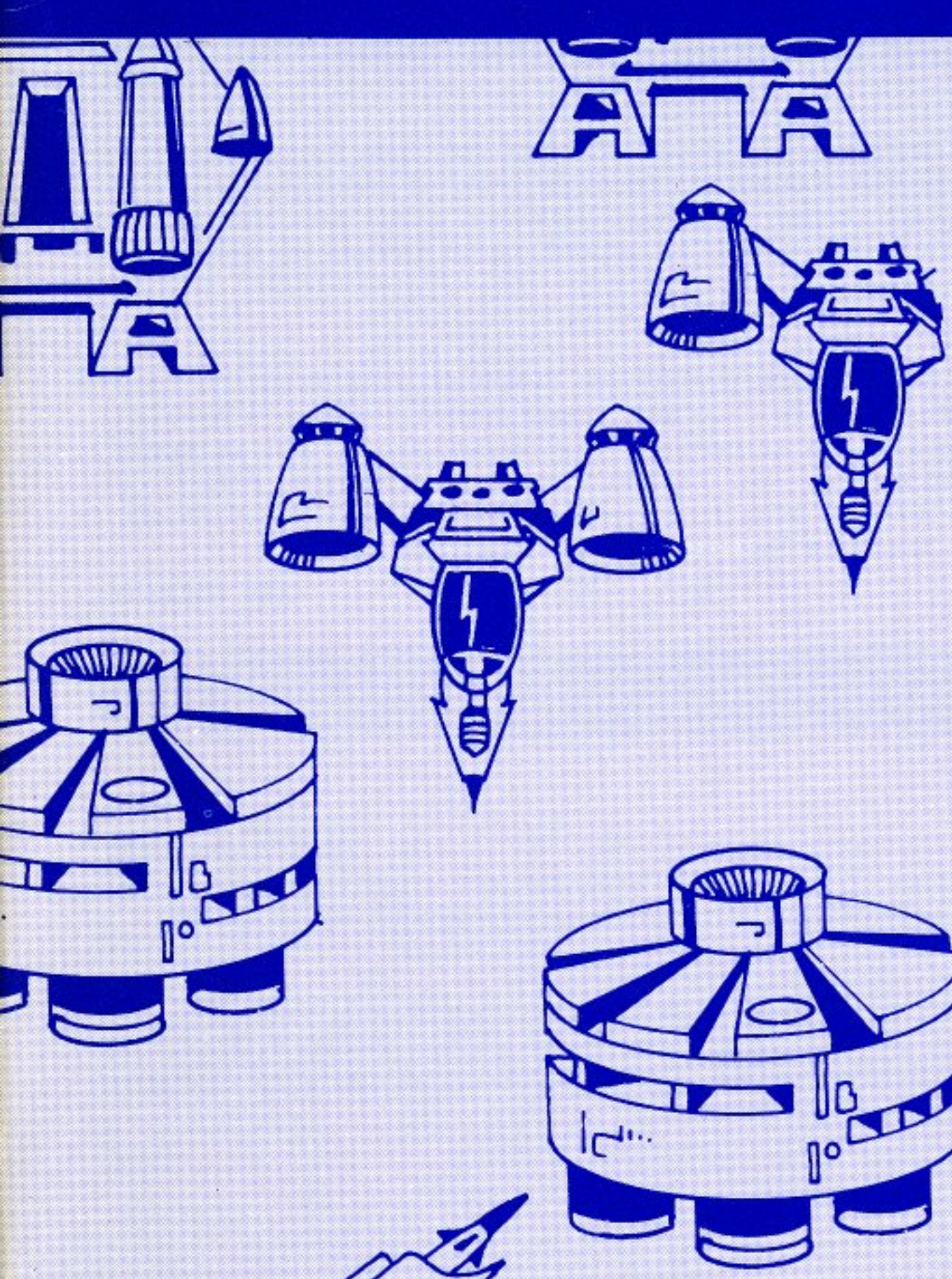
£1.00

ELBUG

FOR THE ELECTRON

Vol 1 No 7 June 1984



Games

* **Play the Stock Market**
* **Flip Flap**

Plus

* **Rotating and Expanding Characters**
* **Screen Freezer**
* **Using the Cursor Keys**

Plus

* **Keyword Abbreviations**
* **Choosing TVs and Monitors**
* **Joystick Interfaces Reviewed**
* **Games Reviews**
* **Hints and Tips**

And much more

GALACTIC INVASION

EDITORIAL

THIS MONTH'S MAGAZINE

One of the most important items of equipment in your microcomputer set up is the TV or monitor that you use with it. The article on choosing a TV or monitor should be helpful if you want to assess the quality of your screen display, and includes some useful test programs.

We have also included the winning Electron entry in the Niagara falls Brainteaser Competition that appeared in the supplement with the March issue. The program produces an excellent screen display, as you will see from the accompanying illustration, every bit as good if not better than the winning entry for the BBC micro. This would surely indicate that ELBUG readers need have no fear of being at a disadvantage in competitions of this kind.

With more utilities and three more high quality games for you to type in and play, this issue of ELBUG should keep you busy and entertained for some time.

ELECTRON ADD-ONS

As promised last month, we are including a review of two of the latest add-ons for the Electron. These both allow the connection of joysticks to the Electron, and both offer excellent value for money. Acorn have just launched their own add-on for the Electron, to be called the Plus One, which we expect to to review in detail in the next issue.

COMPUTER SHOWS

The season for computer shows gets underway with the Computer Fair from the 14th to 17th June at Earls Court, London. Details of other shows appear under the heading of Events in the Supplement. We shall be at many of the London shows, and we would be pleased to welcome any ELBUG readers to our stand - look for the name BEEBUG or BEEBUGSOFT. A visit to a computer show is usually very interesting, and one sure way of seeing all the latest and best products for your micro under one roof.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOAR

HINTS AND TIPS

We thought you might like to know where the £10 and £5 prizes go for the best hints. This month the £10 prize goes to Mrs A.E.Morland and the £5 prize goes to Mr P.Bhandari. We are always on the lookout for more good hints to publish in ELBUG.

MAGAZINE CASSETTE

All the programs featured in this issue of ELBUG are available on cassette, as are all the programs from previous issues. Full details on price and ordering are on the inside back cover. This is certainly the way to avoid all those sore fingers, and you don't have to worry about making mistakes.

ELBUG MAGAZINE

GENERAL CONTENTS

HINTS, TIPS AND INFO

PROGRAMS

TWO JOYSTICK INTERFACES FOR THE ELECTRON

Reviewed by Nigel Harris

In the last issue of ELBUG we made mention of two new interfaces for the Electron to allow the use of joysticks. This time we'll have a closer look at these devices and see what they have to offer.

Product : Joyport, Joystick Interface
Supplier : Signpoint Ltd.,
Unit D, 166a Glyn Road,
Clapton, London. E.5.
Price : £16.95 (inc. VAT & delivery)

Product : Electron S-J Interface
Interface
Supplier : First Byte Computers
10 Castlefields,
Main Centre, Derby DE1 2PE.
Price : £24.95 (inc. VAT & delivery)

The requirement for a joystick facility with a microcomputer is a very common one these days, due largely to the popularity of arcade-action type games programs with many micro users. However, there is no direct connection point on the Electron for a joystick.

Acorn offer a means of expansion, by providing an edge-connector at the back of the machine. This is actually an open set of connections brought out from the circuit board and should be kept covered, when not in use, by the polythene strip provided.

Both of the interface devices in this review fit onto this edge connector and make available the facility to plug in a single joystick. Both are roughly the same size as a packet of cigarettes, and are very light. When plugged in, they are actually supported by the same surface that the host Electron is resting on.

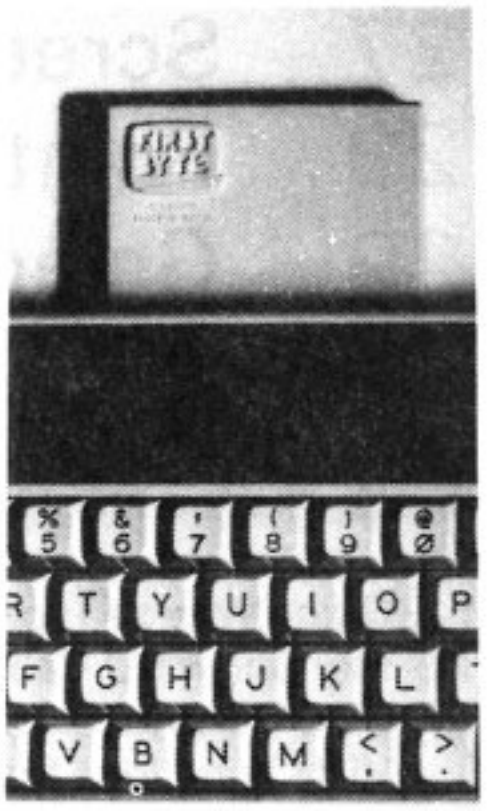
USING EITHER INTERFACE

There are two connections on each interface. One that will fit the computer and one that will fit the joystick. This is a 9 pin 'D' type socket for the standard Atari ,switched type joystick. In other words the joystick can only represent a quantity which is either on or off, or, true or false. Good for direction indication in a program, but not for flying a plane or for drawing with.

Plugging the interface in is best done with the computer switched off. A program (involving some assembler or machine code programming and hopefully supplied and fully documented) must then be loaded and run to set up the function of the joystick with the computer. Each direction of movement of the stick is then interpreted by the computer as a character from the keyboard. Likewise with the fire buttons.

FIRST BYTE S-J INTERFACE

The First Byte interface comes in a glossy, little cardboard package that is going to stack very neatly on shop shelves. The actual device is a similar creamy colour to the Electron. Provided with it are two programs on tape cassette. Both are for configuring the joystick to work with a program that you want to use it with. The first of these gives you a menu of popular games. After you've selected one of them, the program sets up the computer to recognize the particular keys needed by the game in question.



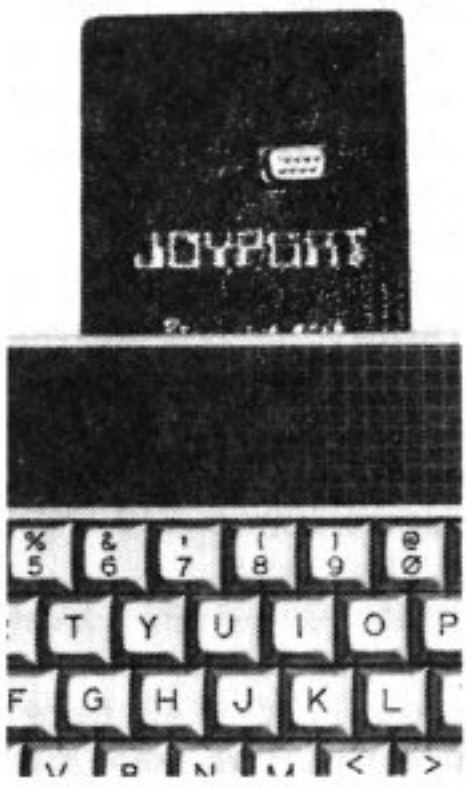
In this way, some twenty different games are catered for, but for your own customising, another program called 'SETUP' will get the computer to recognize your own selection of keys. This is a particularly useful utility, as it makes use of a simple screen diagram and prompts you for the

necessary information. It does not require any technical knowledge of the user, other than knowing which keys are needed in the application. You then load and run the program that you intend to use.

Instructions on use of the tape and of the joystick interface itself are fairly scant. What there is, is quite clear and concise and very helpful, but not adequate, particularly in the case of the interface. Instructions appear on the bottom of the carton and despite a few lines of Basic that help to make use of a joystick from within a Basic program, I still find myself a bit confused about how exactly I should go about obtaining the effect that I want.

THE SIGNPOINT JOYPORT

It is evident from appearance alone that this is the least expensive of the two devices reviewed here. The physical design is certainly not as attractive as First Byte's. Housed inside a thin black plastic case, this one may not take a great deal of that inevitable battering it's going to come in for with kids.



On connecting it to the Electron, it becomes obvious that it's possible to put on upside down. Not that the thinking owner is likely to, but it is possible and it would be enough to cause me some concern for my Electron's safety.

On our review sample, the spacer in the edge connector fixed itself in the slot in the Electron's circuit board repeatedly and had to be prised free with pliers. This was of some inconvenience and it would be very easy to loose this little piece of plastic if care was not taken. Without that of course, the interface could plug into the Electron anywhere that it wanted to along the length of the connector, which again would be hazardous.

At the other end (well actually coming out of the top of the Joyport), the 9 pin D-type socket was mounted directly to the interface's printed circuit board. This also may not be very reliable in the long term, as the solder connections must take the strain of insertion and extraction of the joystick plug.

A cute little Xeroxed booklet came inside the box which simply showed a few examples of how the Joyport interface might be used in Basic programs. These are very useful and in fact every Electron owner could experiment with them as they are all for customising the Welcome tape programs for use with a joystick. Secondly, a photocopied listing of an assembler program was supplied which when run made it possible to customise a commercial arcade game. Having comments written against the program instructions makes it possible for the advanced or interested user to experiment with this themselves and is of general interest anyway. However, the effort to type this in and save it oneself must be made, and it will be a relief to the owner that the program is quite short.

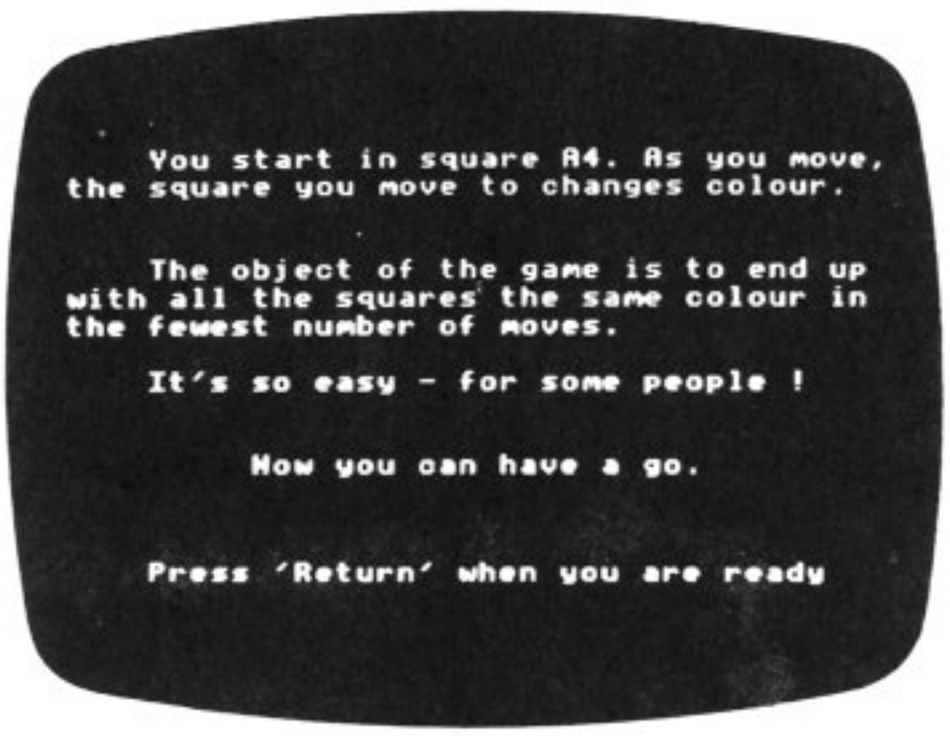
CONCLUSIONS

Neither of these devices allow any further extension of the Electron's expansion bus from its rear connector. That means that both must fit at the end of such an expanded bus. However, both rely on the support of the working surface to a large degree, for the robustness of their link with the host micro. It is hard to imagine how this particular joystick interface design will maintain a reliable connection with an Electron, which has already been expanded. For an extra eight pounds for the First Byte interface you are getting less documentation, a much more elegant design, a cassette with 2 short programs on, convenience for the first time user and much nicer packaging. Whether the extra 50% is justified is a point of personnal preference.

FLIP-FLAP

by S. Goodhead

This is 'a game of thought and strategy', as they say, and is best considered as a board game. The board is square and made up of smaller squares. The player may move between these in one of four directions (up, down, left or right), within the confines of the board. Every move will 'flip' the colour of the small square moved to, from one colour to another. The object is to render a square board, which, is at first randomly filled with two colours, into just one. The challenge is to do this in as few moves as necessary.



The board begins with its squares set to a random mix of both colours and it's then necessary to travel the board in a strategic fashion to revert all squares to just one colour in as few moves as possible. Important points to note however, are that you are not allowed to retrace your last move, and some concentration is required in order that you shouldn't lose track of your last position.

The computer version of this game represents the board on the screen in two colours and prompts for your moves from the keyboard. The following keys are used to make a move:

U	to go U-p
D	to go D-own
L	to go L-eft
R	to go R-ight

and you will see the squares change colour as you move about the board. You won't be able to make any illegal moves and all the moves that you do make will be tallied.

The nice thing about this game is that it's as easy for several people to participate in as it is for one to play - provided you take it in turns! The game has many of the virtues of 'Patience', except that as far as we know, it's not possible to lose; the winner is the one who takes the least moves to complete the board.

PROGRAM NOTES

The program is quite straightforward in its construction (though it resorts to the use of dreaded GOTO instructions in many instances), and not very long. The first few lines set up the computer for running the rest of the program. The rest of it breaks down into the following sections:

Lines 200-250 give the player optional instructions.

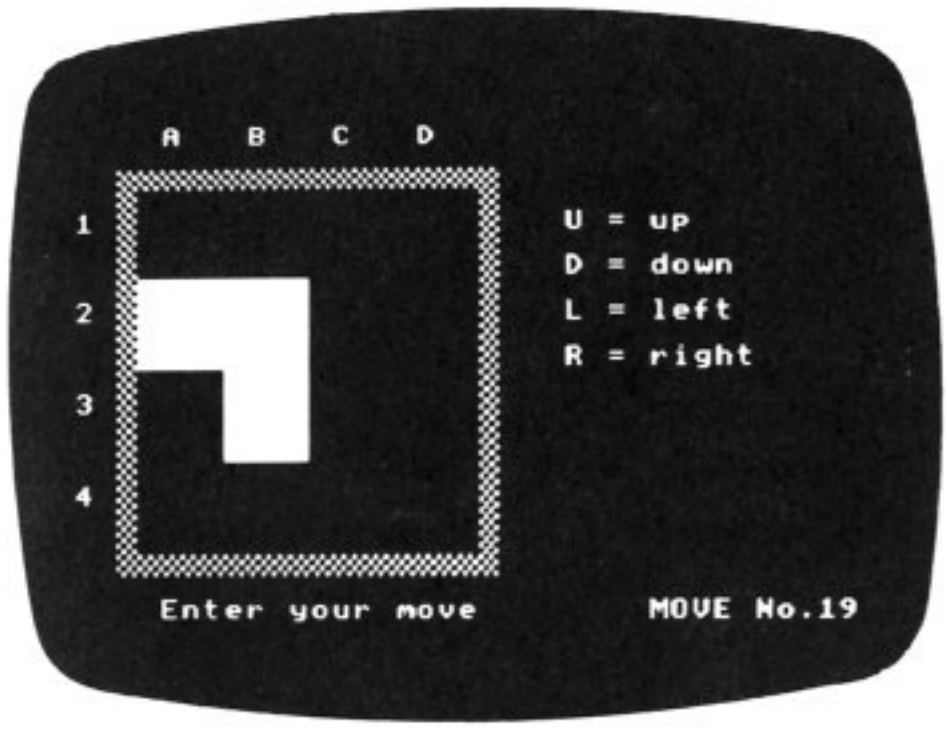
Lines 260-560 construct the board display on the screen.

Lines 570-720 take a move from the keyboard, check for its validity and report back to the player on the screen.Lines 740-830 deal with the end of the game.

Lines 1000-1080 take care of an invalid backwards move.

Lines 1100-1140 take care of an invalid move off the edge of the board.

```
   10 REM Program FLIP-FLAP
   20 REM Version E1.3
   30 REM Author S.Goodhead
   40 REM ELBUG June 1984
   50 REM Program subject to Copyright
   60 :
  100 ON ERROR GOTO 880
  110 *FX4,2
  120 DIM Z$(10):DIM Z%(17,17):DIM A(4,
4)
  130 MODE6
  140 VDU19,0,4,0,0,0
  150 PRINTTAB(0,5)
  160 PRINTTAB(9)"* F L I P - F L A P *
"TAB(18,8)"by"TAB(14,10)"S. Goodhead"
  170 INPUTTAB(3,16)"Would you like ins
tructions ? (Y/N)"X$
```




```
180 IF LEFT$(X$,1)="N" GOTO 280
190 IF LEFT$(X$,1)<>"Y" GOTO 170
  200 CLS
  210 REM Instructions
  220 PRINTTAB(4,5)"You start in square
 A4. As you move,the square you move to
 changes colour."
  230 PRINTTAB(4,9)"The object of the g
ame is to end up with all the squares t
he same colour in the fewest number of
moves."
  240 PRINTTAB(4,13)"It's so easy - for
 some people !"
  250 PRINTTAB(4,16)SPC(5)"Now you can
have a go."'
  260 INPUTTAB(4,20)"Press 'Return' whe
n you are ready"X%
  270 REM Draw the board
  280 MODE4:VDU19,0,4,0,0,0
  290 VDU23,1,0;0;0;0;
  300 VDU23,240,&FF,&FF,&FF,&FF,&FF,&FF
,&FF,&FF
  310 VDU23,241,&CC,&CC,&33,&33,&CC,&CC
,&33,&33
  320 R$=STRING$(4,CHR$240)
  330 S$=CHR$(10)+STRING$(4,CHR$(8))
  340 R$=R$+S$+R$+S$+R$+S$+R$
  350 G$=STRING$(4," ")
  360 G$=G$+S$+G$+S$+G$+S$+G$
  370 FOR X%=5 TO 17 STEP 4
  380 FOR Y%=5 TO 17 STEP 4
  390 P%=RND(2)
  400 IF P%=1 T$=R$
  410 IF P%=2 T$=G$
  420 PRINTTAB(X%,Y%) T$
  430 NEXT:NEXT
  440 PRINTTAB(6,2)"A   B   C   D"
  450 PRINTTAB(4,4)STRING$(18,CHR$241)
  460 X%=4:FOR Y%=5 TO 21:PRINTTAB(X%,Y
%) CHR$241:NEXT
  470 X%=2:PRINTTAB(X%,6)"1"TAB(X%,10)"
2"TAB(X%,14)"3"TAB(X%,18)"4"
  480 X%=21:FOR Y%=5 TO 21:PRINTTAB(X%,
Y%)CHR$241:NEXT
  490 PRINTTAB(4,21)STRING$(18,CHR$241)
  500 FOR X%=5 TO 17 STEP 4
  510 FOR Y%=5 TO 17 STEP 4
  520 READ Z%(X%,Y%)
  530 NEXT:NEXT
  540 RESTORE 1150
  550 X%=5:Y%=17:Q$="O":C=0
  560 PRINT:PRINTTAB(6)"Enter your move
       move no."
  570 PRINTTAB(25,6)"U = up"TAB(25,8)"D
 = down"TAB(25,10)"L = left"TAB(25,12)"
R = right"
  580 REM Get the next valid move
  590 REPEATI$=CHR$(GET)
  600 UNTILI$="U" OR I$="D" OR I$="L" O
R I$="R"
  610 IF I$="U" Y%=Y%-4:IF Y%<5 Y%=Y%+4
:PROCwall:GOTO590
  620 IF I$="D" Y%=Y%+4:IF Y%>17 Y%=Y%-
4:PROCwall:GOTO 590
  630 IF I$="L" X%=X%-4:IF X%<5 X%=X%+4
:PROCwall:GOTO 590
  640 IF I$="R" X%=X%+4:IF X%>17 X%=X%-
4:PROCwall:GOTO 590
  650 IF I$=Q$ PROCback:GOTO 590
  660 IF I$="U" THEN Q$="D"
  670 IF I$="D" THEN Q$="U"
  680 IF I$="L" THEN Q$="R"
  690 IF I$="R" THEN Q$="L"
  700 IF ?Z%(X%,Y%)=255 T$=G$ ELSE T$=R$
  710 PRINTTAB(X%,Y%)T$
  720 C=C+1
  730 PRINTTAB(29,23)"MOVE No.";C
  740 REM Check for end of game
  750 B=0:R=0
  760 FOR M%=5 TO 17 STEP 4
  770 FOR N%=5 TO 17 STEP 4
  780 IF ?Z%(M%,N%)=255 THEN B=B+1
  790 IF ?Z%(M%,N%)=0 THEN R=R+1
  800 NEXT:NEXT
  810 IF B=16 OR R=16 PRINTTAB(5,26)"Ga
me completed in ";C" moves" ELSE GOTO 5
90
  820 INPUTTAB(5,28)"Do you want anothe
r game (Y/N)";X$
  830 IF LEFT$(X$,1)="Y" GOTO 130
  840 MODE6
  850 *FX4,0
  860 END
  870 :
  880 ON ERROR OFF:MODE6
  890 *FX4,0
  900 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
  910 END
  920 :
 1000 REM Illegal retrace move
 1010 DEFPROCback
 1020 IF I$="U" Y%=Y%+4
 1030 IF I$="D" Y%=Y%-4
```

```
 1040 IF I$="L" X%=X%+4
 1050 IF I$="R" X%=X%-4
 1060 VDU7
 1070 PRINTTAB(5,28)"You can not go bac
kwards":FOR D=1 TO 5000:NEXT:PRINTTAB(5
,28)SPC(35)
 1080 ENDPROC
 1090 :
 1100 REM Illegal move into wall
 1110 DEFPROCwall
 1120 VDU7
 1130 PRINTTAB(5,28)"You cant go throug
h the wall !":FOR D=1 TO 2500:NEXT:PRIN
TTAB(5,28)SPC(30)
 1140 ENDPROC
 1150 :
 1160 REM Board data
 1170 DATA&5E68,&6368,&6868,&6D68,&5E88
,&6388,&6888,&6D88,&5EA8,&63A8,&68A8,&6
DA8,&5EC8,&63C8,&68C8,&6DC8
```

BASIC KEYWORD ABBREVIATIONS & TOKENS

Although the Electron keyboard allows a number of Basic keywords to be entered by single keys on the keyboard, in fact, nearly all the Basic keywords can be entered in an abbreviated form, which can be useful in a number of ways as well as reducing the amount of typing required. For example, keyword abbreviations are very useful in function key definitions to save space. The table below lists all the Basic keywords and their abbreviated forms together with the token for each keyword. This is the hexadecimal number which is used to represent each keyword internally.

The keywords marked with an asterisk (*) are those which can be entered by single key strokes from the keyboard. The third column gives the hexadecimal value of the token. Note that the abbreviation includes the opening bracket if one is required.

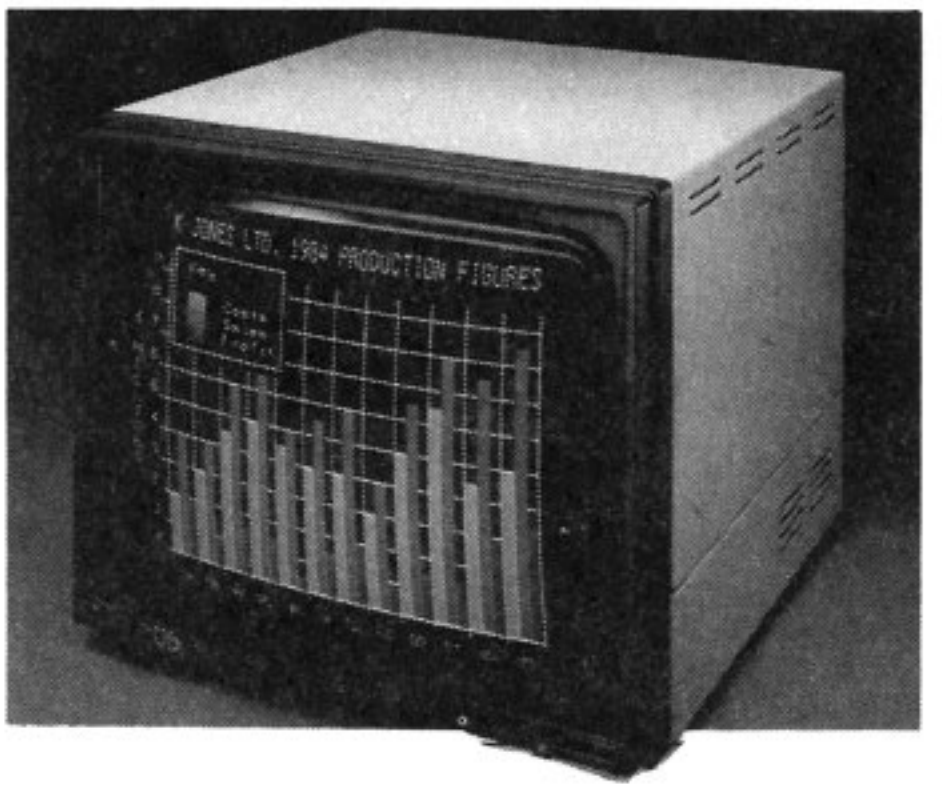
ABS	ABS	94
ACS	ACS	95
ADVAL	AD.	96
AND	A.	8Ø
ASC	ASC	97
ASN	ASN	98
ATN	ATN	99
* AUTO	AU.	C6
BGET	B.	9A
BPUT	BP.	D5
CALL	CA.	D6
* CHAIN	CH.	D7
CHR$	CHR.	BD
CLEAR	CL.	D8
CLG	CLG	DA
CLOSE	CLO.	D9
CLS	CLS	DB
* COLOUR	C.	FB
COS	COS	9B
COUNT	COU.	9C
DATA	D.	DC
DEF	DEF	DD
* DEG	DEG	9D
DELETE	DEL.	C7
DIM	DIM	DE
DIV	DIV	81
* DRAW	DR.	DF
* ELSE	EL.	8B
* END	END	EØ
ENDPROC	E.	E1
ENVELOPE	ENV.	E2
EOR	EOR	82
EOF	EOF	C5
ERL	ERL	9E
ERR	ERR	9F
ERROR	ERR.	85
EVAL	EV.	AØ
EXP	EXP	A1
EXT	EXT	A2
FALSE	FA.	A3
FN	FN	A4
* FOR	F.	E3
GCOL	GC.	E6
GET	GET	A5
GET$	GE.	BE
GOSUB	GOS.	E4
* GOTO	G.	E5
HIMEM	H.	93 (right)
HIMEM	H.	D3 (left)
IF	IF	E7
INKEY	INKEY	A6
INKEY$	INK.	BF
* INPUT	I.	E8
INSTR(	INS.	A7
INT	INT	A8
LEFT$(	LE.	CØ
LEN	LEN	A9
LET	LET	E9
LINE	LIN.	86
* LIST	L.	C9
LN	LN	AA
* LOAD	LO.	C8
* LOCAL	LOC.	EA
LOG	LOG	AB
LOMEM	LOM.	92 (right)
LOMEM	LOM.	D2 (left)
MID$(	M.	C1
MOD	MOD	83
* MODE	MO	EB
MOVE	MOV.	EC
NEW	NEW	CA
* NEXT	N.	ED
NOT	NOT	AC
OFF	OFF	87
* OLD	O.	CB
ON	ON	EE
OPENUP	OP.	AD*
OPENOUT	OPENO.	AE
OPT	OPT	
OR	OR	84
OSCLI	OSC.	FF **
PAGE	PA.	9Ø (right)
PAGE	PA.	DØ (left)
PI	PI	AF
* PLOT	PL.	FØ
POINT(	PO.	BØ
POS	POS	B1
* PRINT	P.	F1
* PROC	PRO.	F2
PTR	PT.	8F (right)
PTR	PT.	CF (left)
* RAD	RAD	B2
READ	REA.	F3
REM	REM	F4
* RENUMBER	REN.	CC
* REPEAT	REP.	F5
REPORT	REPO.	F6
* RESTORE	RES.	F7
RETURN	R.	F8
RIGHT$(	RI.	C2
RND	RND	B3
* RUN	RUN	F9
* SAVE	SA.	CD
SGN	SGN	B4
SIN	SIN	B5
SOUND	SO.	D4
SPC	SPC	89
SQR	SQR	B6
* STEP	S.	88
STOP	STO.	FA
STR$	STR.	C3
STRING$(	STRI.	C4
TAB(	TAB(	8A
TAN	T.	B7
* THEN	TH.	8C
TIME	TI.	91 (right)
TIME	TI.	D1 (left)
TO	TO	B8
TRACE	TR.	FC
TRUE	TRUE	B9
* UNTIL	U.	FD
USR	USR	BA
VAL	VAL	BB
* VDU	V.	EF
VPOS	VP.	BC
WIDTH	W.	FE

CHOOSING TVS AND MONITORS

by Steve Fox

A bewildering range of colour monitors is now available. In this article, Steve Fox offers advice to those who are contemplating a purchase or would like to assess the monitor they are already using, with the test programs provided. We will be reviewing some of the currently available monitors next month.

All good colour displays require a monitor having an RGB input. The computer generates three simultaneous TV signals: red, blue and green. These are fed (usually via a DIN plug) down three separate conductors in a multiway cable to the monitor. The three signals are applied to the picture-tube to generate three superimposed images. The colour you see depends on which combination of the three signals is "on". Thus the red image superimposed on the green image is seen as yellow. Green and blue are seen as cyan, and so forth.



The main difference between the image from a computer and a TV picture is that the computer's signals have only two values, on and off, whereas a TV picture will contain many intermediate tones and shades of colour. If you could get a TV picture by means of an RGB connection, you would have a superb image compared with what you usually see. This is actually done in TV studios, but for transmission the three signals have to be combined into one ("coded"), and then modulated on to a radio-frequency carrier. The TV receiver demodulates the combined signal and then decodes it (i.e. separates it into its three component parts). These transmission processes degrade the signal and your computer display will be similarly degraded if you connect it to a TV receiver by means of the aerial feed.

However, it should not be assumed that a purpose-built RGB monitor will necessarily give you a better picture than a TV receiver which has been modified to take an RGB input. The type of picture-tube in the two is very often identical and so are the video circuits. But of course the TV receiver is enormously more complex containing, as it does, tuner and demodulator stages for vision and sound, a decoder to separate out the red, green and blue signals, audio stages, a loudspeaker and very often remote control also. The omission of all this technology should, by rights, knock some 40% off the price. In practice, a standard TV receiver costs about the same as a dedicated computer monitor of the same size. One has to pay a little more for the TV receiver to be modified for RGB input, but this may prove to be an excellent investment if you wish your monitor to double up as a colour receiver, for example as a portable. So far as I know, no manufacturer actually carries out such modification. The work is done by specialist dealers who usually work to a specification which has been approved by the manufacturer.

Turning now to purpose-built RGB monitors, these come in three grades, depending on the resolution of the picture-tube. The "standard" resolution tubes are from the same range of mass-produced tubes which are found in TV sets. "Medium" resolution tubes are sold in much smaller quantities and so cost a lot more. A typical monitor incorporating such a tube would cost an extra £120. If you want a "High"

resolution tube this will cost another £120 or so. Are these extra costs justifiable? In terms of resolution, the answer is no. Mode 0 graphics are low definition compared with a TV camera and a standard TV tube is more than adequate to resolve any detail which the BBC or Electron micro is capable of generating. But this is not the entire answer.

For various reasons, we tend to view a computer monitor from a distance of 2 or 3 feet, less than half the normal viewing distance for television. As we move in, the screen structure becomes more and more noticeable. If you have never done so, try taking a magnifying glass to the screen of a colour TV receiver: it's a revelation! The structure is particularly distressing when viewing an 80-column word-processor screen, because the distance between the colour stripes ("slot pitch") is comparable with the width of the letters. But before you decide that you need a high resolution monitor to cope with word-processing, do bear in mind that a good monochrome monitor can be had for less than £100 and that its screen structure is faultless!

Of course, there is no doubt that the higher resolution (I prefer to call them "finer structure") screens are very pleasant to use for coloured graphics and if you can afford the luxury, go ahead. Those who can't afford it can get a cost-free improvement by sitting a bit further away from their screens!

Having made your decision regarding the type of monitor you want to buy, study the adverts in the ELBUG supplements, and other magazines. There are substantial variations in the prices asked for an identical monitor, but I would advise against ordering without a demonstration, however reputable you believe the maker and the retailer to be. To arrange this may not be too easy as many of the more competitive firms deal mainly by mail-order, perhaps from a remote part of the country, but you may find it worth persevering. Computer clubs can also be a rich source of wisdom, and members are usually very willing to give a demonstration of their own monitor.

TESTING YOUR TV OR MONITOR

Finally, you can test out your TV or monitor with the programs listed here. The testing and setting-up of monitors is a subject for the expert but, armed with programs 1-4, you can spot most of the faults. Here are some things to watch for.

1. Using Program 1, the red, blue and green images should coincide over the whole area of the screen. If they don't, there will be coloured fringes visible on the white grille and dot pattern.

```
10 REM TVMON1
20 MODE 4
30 VDU23,1,0;0;0;0;
40 FORJ%=0TO1279 STEP 64
50 MOVE J%,0
60 DRAW J%,1024
70 NEXT
80 MOVE 1279,0
90 DRAW 1279,1024
100 FORK%=0TO1023 STEP 64
110 MOVE 0,K%
120 DRAW 1280,K%
130 NEXT
140 MOVE 0,1023
150 DRAW 1279,1023
160 FORJ%=32TO1248 STEP 64
170 FORK%=32TO992 STEP 64
180 PLOT 69,J%,K%
190 NEXT,
```

2. Linearity and geometry should be good i.e. no squashing or extension of the grille and all lines free of curvature.

3. When the computer switches a colour on or off, the vertical edge of the waveform should be "clean", i.e. each block of colour in the colour-bar signal (Program 2) should be of uniform brightness. Avoid monitors which give dark or bright vertical lines along the boundaries.

```
10 REM TVMON2
20 MODE 2
30 VDU23,1,0;0;0;0;
40 FORJ%=0TO1120 STEP 160
50 READ K%
60 GCOL 0,K%
70 MOVE J%,0
80 MOVE J%,1024
90 PLOT 85,J%+160,1024
100 MOVE J%,0
110 PLOT 85,J%+160,0
120 NEXT
130 DATA 7,3,6,2,5,1,4,0
```

4. Use Program 3 to form a subjective judgement of definition. The screen structure should not be too obtrusive.

5. RUN Program 4. The corners of the raster should be on the screen and the raster size should not change as the colours change. The white should neutral and not tinged with a colour.

```
10 REM TVMON3
20 MODE 0
30 FORJ%=0TO2558
40 VDUJ%MOD95+32
50 NEXT
```

```
10 REM TVMON4
20 MODE 4
30 REPEAT
40 FORJ%=0TO7
50 VDU19,0,J%;0;
60 Z=INKEY 100
70 NEXT
80 UNTIL 0
```

Some of the defects which may be shown up by these tests may be remediable by the vendor: others will not. To avoid disappointment, don't buy a monitor unless it has actually performed to your satisfaction. Next month we will review some of the monitors now available for the home computer user.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

QUICK WAY TO TYPE OUT THE CONTENTS OF P% - M.Davis

Type the '[' key (Shift and Copy key together) followed by Return and the value of P% is printed in hexadecimal form. This is the contents of the 'program counter'; of interest for assembler programming.

MERGING VDU CALLS - P.Vincent

Don't forget that successive VDU calls may be merged into one, for example,
VDU 12,14,19,1,4,0,0,0 is the same as
VDU 12:VDU 14:VDU 19,1,4,0,0,0

BBC/ELECTRON ACORNSOFT COMPATIBILITY - R. Paine

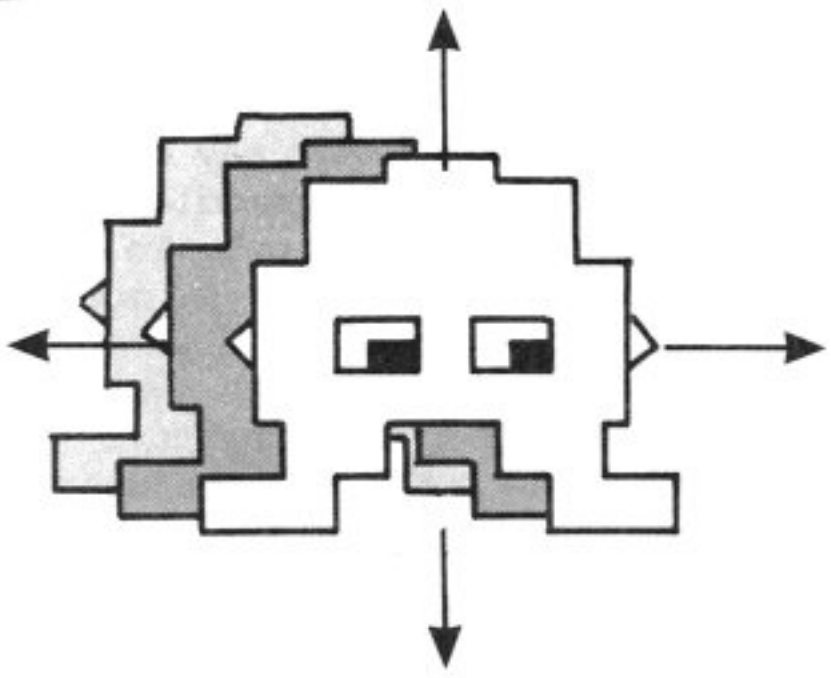
Mr. Paine informs us he has been notified by Acornsoft that the program packages 'BCPL', 'Graphs and Charts', and 'Creative Graphics' for the BBC micro, will run in their entirety on the Electron (provided you do not do something like selecting Mode 7 presumably). There are more packages that will run perfectly well, but with the shortcoming of a distorted title page while loading, (this due to the use of the Mode 7 display for the title page). These other programs include 'Word Sequencing', 'Word Hunt', 'Lisp', 'Sliding Block Puzzles', 'Chess', 'Missing Signs', 'Turtle Graphics', 'Desk Diary', 'Snooker', 'Forth', 'Microtext', 'Picture Maker', and 'Cube Master'. Many of these are or soon will be available especially for the Electron.

USING THE CURSOR KEYS IN A PROGRAM

by Philip Le Grand

You are probably familiar with using the cursor keys for editing purposes but they can also be used within a program to control, for example, vertical and horizontal movement on the screen. This is usually achieved by changing the normal function of the cursor keys so that they generate their equivalent ASCII (American Standard Code for Information Interchange) code, using the command *FX4,1. The particular code values associated with the cursor keys are:

up	139	down	138
left	136	right	137
copy	135		



For example, try running the following program:

```
10 MODE 6
20 *FX4,1
30 A=GET
40 IF A=139 THEN PRINT "UP PRESSED"
50 GOTO 30
```

You should find that any time you press the cursor up key then a corresponding message is displayed on the screen otherwise nothing. The program could easily be extended to test for any of the other cursor keys as well.

In a similar way, the cursor keys can be used as extra function keys using another variant of *FX4 (see the 'Extra Function Keys' hint in ELBUG Vol.1 No.4 and also the User Guide, Appendix D). The normal functions of the cursor keys are reinstated with:

```
*FX4,0 <return>
```

With all types of * command you should ensure that they are the only, or last, instruction in any line of the program.

The following example program demonstrates how the function keys can be detected within a program. It displays an 'invader' on the screen, and then using the four cursor direction keys, the invader can be moved to any part of the screen.

```
100 REM CURSOR KEYS DEMO
110 ON ERROR GOTO 270
120 MODE 1
130 VDU23,244,102,153,60,90,126,60,
66,129
140 VDU 5:GCOL3,2
150 *FX4,1
160 X%=640:Y%=512
170 REPEAT
180 X1%=X%:Y1%=Y%
190 MOVE X%,Y%:PRINT CHR$(244)
200 A%=GET
210 IF A%=136 THEN X%=X%-4
220 IF A%=137 THEN X%=X%+4
230 IF A%=138 THEN Y%=Y%-4
240 IF A%=139 THEN Y%=Y%+4
250 MOVE X1%,Y1%:PRINT CHR$(244)
260 UNTIL FALSE
270 MODE 6:*FX4,0
280 END
```

PROGRAM NOTES

The *FX4 command to turn off normal editing facilities can be found in line 150. These are reset in line 270 if you exit from the program. The other lines, up to line 160 select screen Mode 1, define a character for the invader (line 130), join the text and graphics cursors to allow finer positioning of characters and select the initial position of the invader with X% and Y%.

In the main loop of the program, the invader is first displayed in yellow,

using Exclusive-OR plotting, and then the computer waits for a key to be pressed. When this happens, the ASCII value of the key is returned in A%. Lines 210 to 240 then check to see which, if any, of the cursor keys have been pressed, in manner very similar to our first example. For example, line 210 tests whether the value in A% corresponds to the left cursor key. If it does, then the x co-ordinate is decreased by 4. The invader is replotted at the old X%,Y% position to erase the previous image, and the loop returns to the start to display it in the new position. If any other key is pressed, the invader is redisplayed at the same point. The loop is repeated indefinitely, or until you press Escape or Break.

At the moment, the program will move the invader off the screen if it is moved far enough. However it is possible to prevent this by changing lines 210 to 240 to take account of this:

```
210 IF A%=136 AND X%>10 THEN X%=X%-4
220 IF A%=137 AND X%<1240 THEN X%=X%+4
230 IF A%=138 AND Y%>30 THEN Y%=Y%-4
240 IF A%=139 AND Y%<1000 THEN Y%=Y%+4
```

Line 210 now checks to see if the left cursor key has been pressed AND if the invader is more than 10 graphics units in from the left hand side of the screen before the new position for the invader is calculated.

The reason for using Exclusive-OR plotting is to avoid the need to erase an invader by redefining the invader's colour to black, plotting it at the old position to erase it, and then defining its colour to be yellow again. If you display any object using Exclusive-OR plotting (GCOL 3,n), then plotting the object a second time effectively erases it from the screen.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

VDU FUNCTIONS FROM CONTROL CHARACTERS - Mrs. A.E. Morland

For a blue striped background for your text, without using a function key or writing the command on the screen, keep the CTRL key down and type 'SBD@@@' (but not in Modes 2 or 5).
(What this does is to send the equivalent of a VDU 19,2,4,0,0,0 command to the Electron which redefines the logical colour of the background as blue. See the VDU 19 command in the User Guide p. 266, or for more ideas, see Appendix A. For example CTRL 'L' will clear the screen, or CTRL 'SAB@@@' will make a colour TV look like a green screen monitor).

PROGRAM LENGTH - D.Reisenberger

To find out the length of a program that has been loaded from tape, print the decimal equivalent of the four figure hex number that appeared on the screen at the end of loading. For example, if it is 3400, then typing 'Print &3400' will give the required number - ie.13312 bytes.

QUICK PROC OR FN LOCATE - P.Bhandari

To find a procedure in a very long program without searching through a long listing of it, try this. To find, for example 'PROCintro', type in immediate mode

```
TRACEON:PROCintro
```

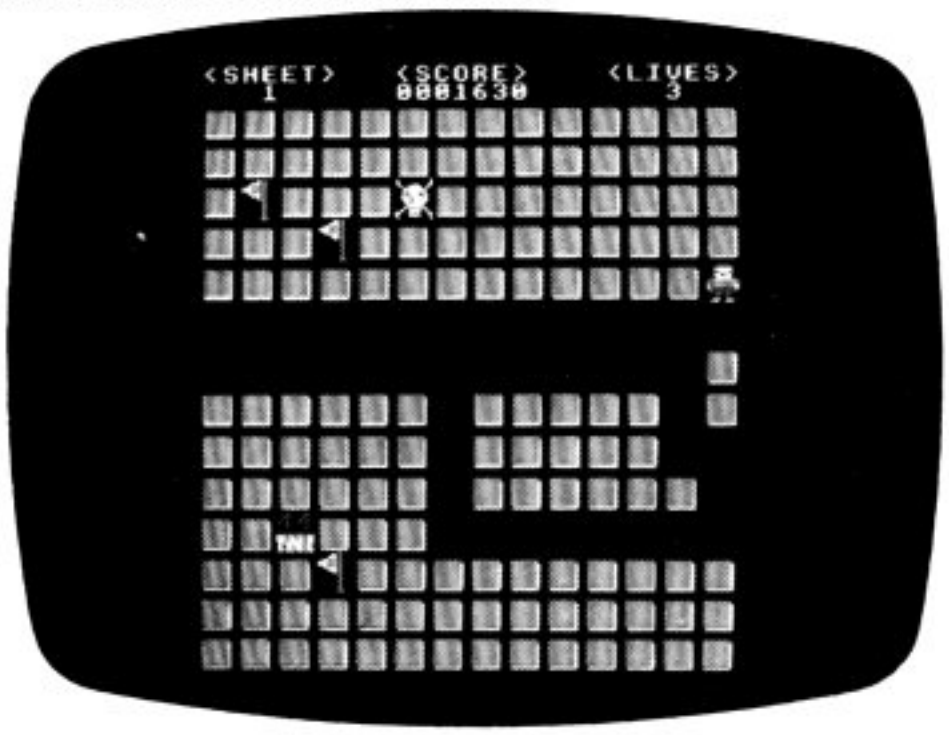
The line number at which the Basic code for PROCintro starts will then be printed on the screen in the first set of brackets. Don't forget to give any parameters that may be required. To do the same with functions, type

```
TRACEON:PRINT FNintro
```

(remembering again to give it any parameters that may be needed).

FOUR GAMES REVIEWED

Title : Danger UXB
Supplier : Program Power
Price : £7.95
Rating : ****
Reviewer : Alan Webster



Do you remember the popular television series "Danger! UXB"? It was about members of a bomb disposal squad defusing unexploded bombs around the time of the second world war. Well, this game of the same name has cast you in the role of the bomb disposal expert.

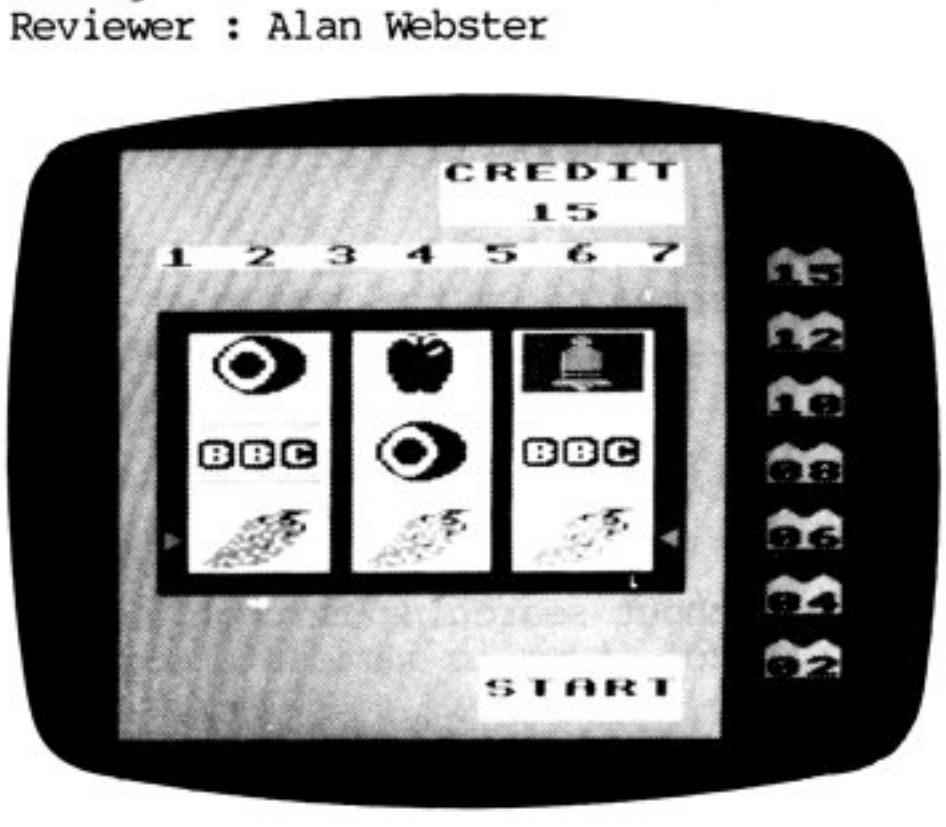
Each bomb is represented by the word TNT and a number which counts down towards zero. If the number reaches zero, the bomb explodes and you lose a life. You can defuse the bomb by running on to the square that contains the bomb.

There are skulls warning of unexploded bombs positioned around the screen, and these must be avoided at all costs. The game is made more difficult as each square on the board can only be visited once per screen, and some thought is needed to work out the best route each time.

The game has three skill levels, with the third being very difficult, as you are being chased by big army boots at the same time as you are trying to defuse the bombs.

This is a very colourful and enjoyable game, and one that you will want to play again and again.

Title : Fruit Machine
Supplier : Superior Software
Price : £7.95
Rating : **
Reviewer : Alan Webster



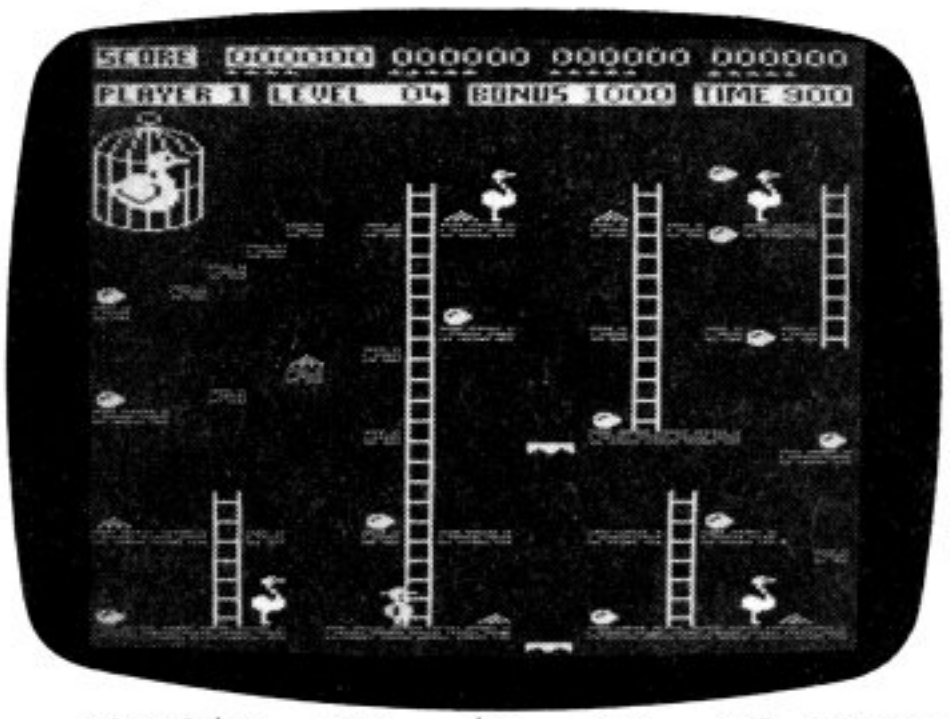
Fruit Machine is a Mode 2 implementation of the arcade and pub favourite, but unfortunately it does suffer quite a lot from lack of speed.

I found the game quite tedious after a while, as you wait for the reels to spin, plus the fact that there are very few extras included in this game, apart from a simple nudge and gamble facility.

Fruit machine games demand the full range of colours available in Mode 2 to provide a really good screen display, but the resulting lack of speed considerably reduces the appeal of this version.

[See our series on "Using BBC Micro Programs on an Electron" in issues 4 to 6 of ELBUG for more information about the effect of screen mode on the speed of the Electron - Ed.]

Title : Chuckie Egg
Supplier : A & F Software
Price : £7.90
Reviewer : David A. Fell
Rating : ****



Chuckie Egg is one of those addictive games that you could happily play all day, given nothing else to do. The scenario is very basic; you are a rather peculiar looking bird called Chuckie Egg, and you have to walk and fly around a complex of ladders, floors, and (later on) elevators, collecting eggs. Each round is completed when all of the eggs have been gathered, and you can gain extra points by eating the piles of grain that are also present. There is a bonus for completing the round as quickly as possible, and hazards are provided in the form of stork-like birds, which eat the piles of grain, and kill you if you get too close, with more features appearing as the game advances.

The game gets increasingly harder as you progress through different levels, and the graphics, although perhaps a little simple, are quite pleasing, as are the sound effects. This game is certainly well worth purchasing, and will provide hours of entertainment to the game player.

Title : Invaders
Supplier : Superior Software
Price : £7.95
Reviewer : David A. Fell
Rating : ***

This game is an implementation on the Electron of the classic arcade game Space Invaders. It runs in Mode 5 (and hence only uses four colours), and features some attractively designed characters. The movement in the game is smooth and flicker free, but it does slow down noticably when you are firing and a 'mothership' crosses the screen. The sound effects are a little uninspired, and the game would probably be better off without some of these; they tend to be a boring, monotonous din on the ear drums after a few thousand points.

I don't feel that I could fully recommend this game to all but the most persistant of games players. If you do play this game, but find the sound effects unpleasant, *FX210,1 typed in before the CHAIN"" will turn off all the sound. (Note: don't press Break, as this will cancel the effect of this.)

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MACHINE CODE AT &E00 - M.Tilley

If a machine code program is required to reside in the page starting at address &E00, preface it with the two empty bytes (at &E00 and &E01) and make &E02 the first address used. If this is not done, pressing the Break key will destroy the first two bytes of the program. (If &0D and &FF are inserted into addresses &E00 and &E01, then the message Bad Program will no longer appear).

BLACK ON WHITE TEXT - George Foot

If you're trying to use an 80 column Mode display and finding it difficult to read, try reversing the foreground and background colours. Black text on a white background is sometimes more legible on a poor display.
(In MODE 0, this can be done by entering VDU19,0,7,0,0,0:VDU19,1,0,0,0,0)

ROTATING AND EXPANDING CHARACTERS

by P. Manhire

This is a collection of short routines in assembly code which will allow any printable character (alphanumeric or user defined) to be rotated or enlarged in any position on the screen.

The routines described here can be used in many ways, for example:

(1) Produce quadruple sized text in all modes.
(2) Producing new character defin- itions (for games etc.). It is possible to produce giant invaders or inverted frogs or rotating munchy-men or whatever, from previous character definitions without having to work out the new definitions.
(3) Labelling the vertical axes of graphs, diagrams, etc.

The program listed at the end of the article is in two parts. The assembly code which manipulates characters is contained in the procedure PROCassemble listed from line 1000 to 1730. It is this procedure that should be added to your own programs to incorporate the rotate and expand routines. The main program, from lines 10 to 260 provides a short demonstration of the use of these routines. Type the whole program in and save it to cassette. You may also wish to save the procedure separately using *SPOOL (delete lines 10 to 260 first) so that you can use *EXEC to append this procedure to any of your own programs. This technique was explained in the article on functions and procedures in ELBUG issue 5.

To use the rotate and expand routines, the procedure PROCassemble must first be called to assemble the machine code at &A00, and this is done in the demonstration program at line 100. This area of memory is not normally used by the Electron, and is thus available to the user. You can change this address by altering line 350 (see Note 1 later). The machine code is subdivided into five routines which may each be called separately. The five calls and their functions are given in the table.

If you run the complete demonstration program, it results in any letter entered from the keyboard being printed to the screen in a variety of ways - reflected, rotated, inverted and expanded. Inspection of this program will show how to call and use the various routines, but here are a few additional examples and notes on their use.

All operations must begin by setting X% equal to the ASCII code of the character to be manipulated (it can be one that you have defined yourself in the range 224 to 255) and then calling Getchr. Use the Electron User Guide, page 285 to get ASCII codes - eg. ASCII 75 = "K". If you want to manipulate your own characters, first define them with VDU23 (see User Guide page 93), and then set X% to the character number that you have chosen (avoid values 250 to 254 since these are used by this program).

To take a real example, we will manipulate a space invader. These steps assume that the procedure PROCassemble has been loaded into memory first and the machine code assembled by typing:

PROCassemble <return>

ASSEMBLER PROCEDURES

Getchr This reads the dot pattern of any character into memory at locations &71 to &78 and &81 to &88. To use it; first set X% to the ASCII code of the character you want to manipulate and then CALL Getchr.

Putchr This performs a VDU 23 call to define a character with the manipulated character in &81 to &88. To use it; first set X% to the ASCII code of the character you want to define.

Rot This rotates the character read by Getchr ANTICLOCKWISE by 90 degrees. CHR$250 is always defined as the rotated character but Rot followed by Put will define any other character.

Invert This inverts the character read by Getchr (ie. like the reflection in a pool of water). CHR$250 is always defined as the inverted character but Invert followed by Put will define any other character.

Expand This takes the character read by Getchr and expands it to fill a block of four characters (always CHR$251 to CHR$254).

Type 'NEW <return>' to delete the unassembled version of the procedure before entering the lines below. Line 10 below creates the space invader as character 224 (try executing VDU224 after this in modes 4 or 5). Line 20 stores the character in the locations reserved by the manipulator routine. To rotate the invader you simply call the routine 'Rot' (in line 30 below). The rotation and inversion routines 'Rot' and 'Invert' define the new character as character 250, so to print it, line 40 is used below:

```
10 VDU23,224,60,126,219,255,126,60,66,36
20 X%=224:CALL Getchr
30 CALL Rot
40 PRINT CHR$250
```

To turn it on its head, add:

```
50 X%=250:CALL Getchr
60 CALL Rot
70 PRINT CHR$250
```

This picks up the rotated invader, and rotates it once again.

The following sequence will produce an expanded version:

```
X%=224:CALL Getchr:CALL Expand
```

This expands the original character into a block of four characters (ASCII 251, 252, 253 and 254) while maintaining the shape.

The easiest way to print the expanded character is to define a variable say 'character$' as follows:

```
character$=CHR$251+CHR$252+CHR$10+CHR$8
+CHR$8+CHR$253+CHR$254
```

The characters with ASCII values 10 and 8 move the cursor down one line and back two positions to complete the bottom two characters making up the block of four. Then typing:

```
PRINT character$ <return>
```

will print the expanded character. This could be positioned anywhere on the screen using PRINT TAB instead of just PRINT.

Here are two further examples:

```
X%=ASC"W":CALL Getchr:CALL Rot:PRINT CH
R$250
```

would print a letter "W" on its side.

```
PRINT"£":X%=ASC("£"):CALL Getchr:X%=250
:CALL Rot:CALL Getchr:CALL Expand: PRIN
T CHR$251;CHR$252'CHR$253;CHR$254
```

would produce:

£" £"

NOTE 1: The code is loaded between &A00 and &AFF. This APPEARS to be safe but if you do not trust this you must

relocate the code elsewhere. This is done by changing line 350 in PROCassemble from P%=&A00 to P%=(new location)

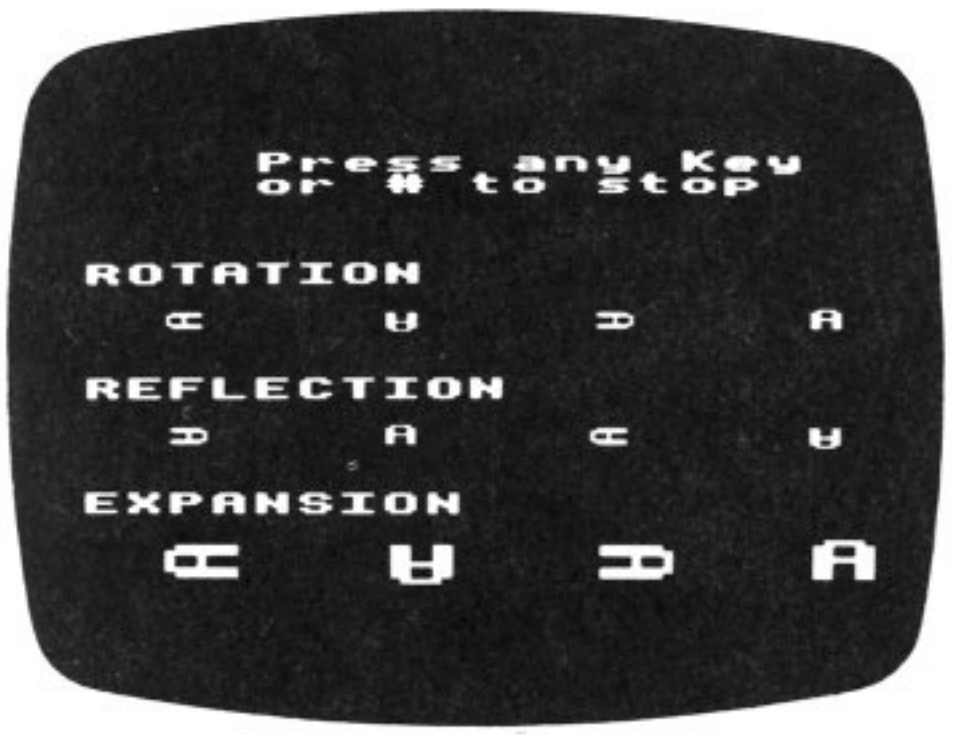
NOTE 2: The code can be saved without the BASIC assembler. This is done by running the program as listed and then typing:

```
*SAVE"CODE" A00 B00 A00 <return>
```

The code will then be saved on tape. To reload it directly into memory type:

```
*LOAD"CODE" <return>
```

and press 'Play' on the cassette recorder. If you do this you will not be able to call the machine code procedures by name (Getchr etc). You must use the addresses in the REM statements in PROCassemble instead. For example, use CALL &A5D instead of CALL Expand.

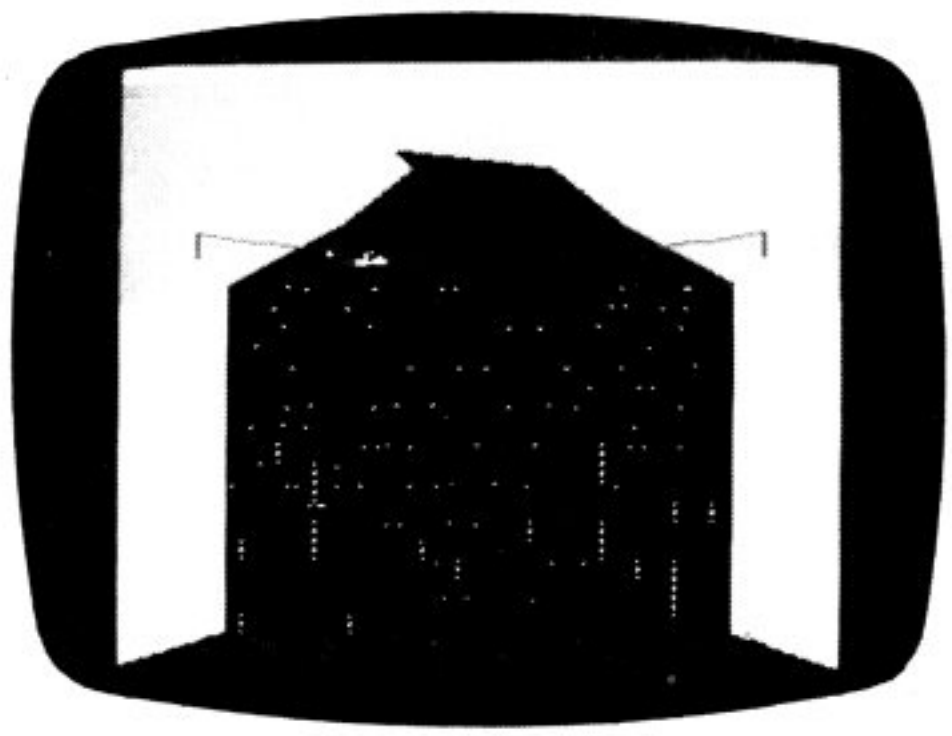



```
  10 REM PROGRAM CHMAN
  20 REM AUTHOR  P.Manhire
  30 REM VERSION E0.1
  40 REM ELBUG   JUNE 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 PROCassemble
 110 A$=CHR$8+CHR$8+CHR$10
 120 B$=CHR$251+CHR$252+A$+CHR$253+CHR
$254
 130 MODE5:VDU19,2,5,0,0,0,19,3,6,0,0,0
 140 PRINT TAB(4,5);"Press any Key":C$
=GET$
 150 IF C$="#" THEN MODE6:END
 160 PRINT TAB(0,10);"ROTATION"
 170 PRINT TAB(0,15);"REFLECTION"
 180 PRINT TAB(0,20);"EXPANSION"
 190 COLOUR2:X%=ASC(C$):CALL Getchr
 200 X%=250:FOR I%=0 TO 3:CALL Rot:PRI
NT TAB(2+I%*5,12);CHR$250:CALL Getchr:N
EXT
 210 X%=250:CALL Getchr:CALL Invert:CA
LL Getchr:FOR I%=0 TO 3
 220 CALL Rot::PRINT TAB(2+I%*5,17);CH
R$250:CALL Getchr:NEXT
 230 X%=ASC(C$):CALL Getchr:X%=250
 240 FOR I%=0 TO 3:CALL Rot:CALL Getch
r:CALL Expand:PRINT TAB(2+I%*5,22);B$:C
ALL Getchr:NEXT
 250 COLOUR3:PRINT TAB(4,6);"or # to s
top":GOTO 140
 260 :
1000 DEFPROCassemble
1010 REM  Assembles code at &A00
1020 REM  CALLS NAME     ADDRESS
1030 REM        Getchr   &A00
1040 REM        Rot      &A17
1050 REM        Putchr   &A33
1060 REM        Invert   &A4D
1070 REM        Expand   &A5D
1080 FOR J%=0 TO 3 STEP 3
1090 P%=&A00
1100 [OPT 0
1110 .Getchr
1120 \****
1130 STX &70:STX &80:LDX#&70:LDY #0:LD
A #10
1140 JSR &FFF1:LDX#&80:LDY #0:LDA #10
1150 JSR &FFF1:RTS
1160 .Rot
1170 \****
1180 LDX #&71:LDY #&88
1190 .Loop:ROL 0,X:TXA:STY &79:LDX &79
1200 ROL 0,X:TAX:INX:CPX #&79:BNE Loop
1210 LDX #&71:DEY:CPY #&80:BNE Loop
1220 LDX #250
1230 .Putchr
1240 \****
1250 LDA #23:JSR &FFEE
1260 TXA:JSR &FFEE
1270 LDX #&81
1280  .Loop2:LDA 0,X:JSR &FFEE
1290 INX:CPX #&89:BNE Loop2:RTS
1300 .Invert
1310 LDX #&71:LDY #&88
1320 \****
1330 .Loop3
1340 LDA 0,X:STA 0,Y
1350 INX:DEY:CPY #&80:BNE Loop3
1360 LDX #250
1370 JMP Putchr
1380 .Expand
1390 \****
1400 LDX #251:STX&8E
1410 LDX #0:STX &8C:STX &8D
1420 LDX #&71:STX &8A
1430 LDX #&81:STX &8B
1440 JSR Qchr
```

```
1450 LDX #252:STX&8E
1460 LDX #0:STX &8C:STX &8D
1470 LDX #&71:STX &8A
1480 LDX #&81:STX &8B:JSR Qchr
1490 LDX #253:STX&8E
1500 LDX #0:STX &8C:STX &8D
1510 LDX #&75:STX &8A
1520 LDX #&81:STX &8B :JSR Qchr
1530 LDX #254:STX&8E
1540 LDX #0:STX &8C:STX &8D
1550 LDX #&75:STX &8A
1560 LDX #&81:STX &8B :JSR Qchr
1570 RTS
1580 .Qchr
1590 .Loop5
1600 LDX &8A:ROL 0,X:LDX &8B:JSR Enter
1610 LDX &8C:INX:STX &8C:CPX #4
1620 BNE Loop5:LDX #0:STX &8C:LDX &8A
1630 INX:STX &8A:LDX &8B:INX:INX
1640 STX &8B:LDX &8D:INX:STX &8D
1650 CPX #4:BNE Loop5:LDX &8E:JSR Putc
hr
1660 RTS
1670 .Enter:BCS Full
1680 CLC:ROL 0,X:CLC:ROL 0,X:INX
1690 CLC:ROL 0,X:CLC:ROL 0,X:RTS
1700 .Full
1710 SEC:ROL 0,X:SEC:ROL 0,X:INX
1720 SEC:ROL 0,X:SEC:ROL 0,X:RTS
1730 ]:NEXT:ENDPROC
```

NIAGARA FALLS BRAINTEASER WINNER

In the supplement included with the March issue of ELBUG, we presented a Brainteaser competition on the theme of crossing Niagara Falls. The winning Electron entry was from Kevin Allen who has now received his £20 prize, and the screen display produced by his winning program is shown in the illustration. Unfortunately, this cannot show the man actually moving along the rope, nor the falling water. We list here Kevin's winning program.



```
  10 REM PROGRAM NIAGARA
  20 REM AUTHOR  Kevin Allen
  30 REM ELBUG  JUNE 1984
  40 :
 100 MODE 2:PROCsetup:PROCscreen
 110 REPEAT
 120 PROCwater:VDU19,10,I%;0;
 130 IF I%=4:I%=7:ELSE I%=4
  140 VDU 19,9,I%;0;
  150 SOUND 0,-15,4,6+RND(4):PROCman
  160 UNTIL FALSE
  170 :
 1000 DEF PROCman:X%=X%+32
 1010 IF X%>1224:MOVE X%-32,716:PRINTMA
N$:X%=32:MOVE X%,716:PRINTMAN$:ENDPROC
 1020 IF X%<192 OR X%>1024:MOVE X%-32,7
16:PRINT MAN$:MOVE X%,716:IF MAN$=MAN2$
:MAN$=MAN1$:PRINT MAN$:ENDPROC
 1030 IF X%<192 OR X%>1024:IF MAN$<>MAN
2$:MAN$=MAN2$:PRINT MAN$:ENDPROC
 1040 X=(X%-576)*2:Y%=((141.42*(EXP(A*X
)+EXP(-A*X)))-(141.42))DIV2+616
 1050 IF X%=192:y%=716
 1060 MOVE X%-32,y%:PRINT MAN$
 1070 MAN$=MAN3$
 1080 MOVE X%,Y%:PRINT MAN$:y%=Y%
 1090 ENDPROC
 1100 :
 1110 DEF PROCwater
 1120 VDU 4,28,3,29,16,12
 1130 SOUND 0,-15,4,RND(3)+6
 1140 PRINT TAB(RND(14)-1,8);CHR$(240);
TAB(RND(14)-1,7);CHR$(240);TAB(RND(14)-
1,7);CHR$(240)
 1150 VDU30,11,28,7,29,12,12,30,11,26,5
 1160 GCOL 0,7
 1170 TIME=0:REPEAT:PLOT 69,RND(832)+19
2,632+RND(4):UNTIL TIME>7
 1180 ENDPROC
 1190 :
 1200 DEF PROCsetup
 1210 VDU 23,1,0;0;0;0;
 1220 VDU 23,240,16,0,0,16,16,0,0,16
 1230 VDU 23,241,255,60,96,0,96,60,255,
96
 1240 VDU 23,242,0,0,12,18,33,66,128,0
 1250 VDU 23,243,16,16,40,68,131,0,0,0
 1260 VDU 23,244,0,16,28,24,16,0,0,32
 1270 VDU 23,245,60,0,0,0,0,0,0,24
 1280 VDU 23,246,0,8,0,0,0,56,56,24
```

Continued on Page 33 ⟶

ELECTRON GRAPHICS (Part 7)

by Mike Williams

This month, in our series on Electron Graphics, we shall look at some more of the many variations that are possible with the PLOT instruction, that has already provided us with several very useful procedures.

So far, we have looked at two very specific uses of the PLOT instruction in Basic, the use of PLOT85 to display solid looking triangles, and PLOT77 which we can use to fill or colour any shape. At this stage it is useful to realise that PLOT is a general purpose graphics instruction unlike the MOVE and DRAW instructions that we looked at in an earlier article. In fact the general instruction PLOT0,X,Y is exactly equivalent to MOVE X,Y, and PLOT5,X,Y is exactly the same as DRAW X,Y.

RELATIVE PLOTTING

In the past, whenever we have wanted to refer to a point on the screen, we have specified a position (X,Y) on the screen where X and Y are the horizontal and vertical distances from the origin (0,0). This is called an absolute reference or position. The point (0,0) is normally at the bottom left hand corner of the screen, though you can change this, as we did for circle plotting, using the VDU29 instruction. For example, you may remember the procedure we wrote for drawing a square:

```
1000 DEF PROCsquare(size,x,y)
1010 MOVE x,y
1020 DRAW x,y+size
1030 DRAW x+size,y+size
1040 DRAW x+size,y
1050 DRAW x,y
1060 ENDPROC
```

There is a second way of specifying a point on the screen, by giving the distance from the last point visited by the graphics cursor, rather than the absolute or fixed position. This uses PLOT1 instead of DRAW (or PLOT5). If we rewrite the procedure above, it will read as follows:

```
1000 DEF PROCsquare(size,x,y)
1010 PLOT 0,x,y
1020 PLOT 1,0,size
1030 PLOT 1,size,0
1040 PLOT 1,0,-size
1050 PLOT 1,-size,0
1060 ENDPROC
```

If you type in either of these procedures, remember that you can test them out in immediate mode, provided you select a graphics mode first, by just typing PROCsquare followed by suitable parameters in brackets. For example:

```
MODE 4          <return>
PROCsquare(200,640,512)     <return>
```

This will draw a square of side 200 with its bottom left hand corner in position (640,512).

In the second version of the procedure, I have used PLOT instructions throughout, although I could just as easily have used a MOVE instruction at line 1010 instead. The instructions used here to draw the square could be written as follows: "from the starting point, draw a distance 'size' vertically upwards, then a line the same distance horizontally, then another line the same distance but now downwards (hence the minus sign), and lastly a fourth line the same distance again back to the starting point (to the left, and thus again minus)". At no time do we use the actual position of any corner of the square as everything is specified relatively.

Whether you prefer the first or the second method is, I think, largely a matter of personal choice, though some might argue that the second method involves less arithmetic, and is thus both simpler and faster. Just decide which method you prefer to use, but also be guided by the circumstances and always try to choose the simplest method if a choice is available.

DOTTED LINES OR SOLID LINES

Another option available with the PLOT instruction is the drawing of dotted instead of the more usual solid lines. Adding 16 to the 'PLOT' number will produce a dotted rather than a solid line. Thus PLOT17 is the same as PLOT1 (relative plotting) but with a dotted line, and PLOT21 is likewise similar to PLOT5 (absolute plotting).

Although a PLOT instruction is normally listed with a specific 'PLOT' number, this can instead be replaced by a variable. For example, we will rewrite the last procedure so that it will draw a square with either a solid or a dotted line. We will do this by introducing an extra parameter, which I'll call 'type', which will have the value 0 for a square of solid lines, and 1 for a square with dotted lines.

The revised version of our procedure is as follows:

```
1000 DEF PROCsquare(type,size,x,y)
1010 LET t=16*type+1
1020 PLOT 0,x,y
1030 PLOT t,0,size
1040 PLOT t,size,0
1050 PLOT t,-size
1060 PLOT t,-size,0
1070 ENDPROC
```

Remember that if you type this in, you can test it out in immediate mode. Thus:

PROCsquare(0,400,200,200)

would give a square of size 400 and in position (200,200) drawn with solid lines while

PROCsquare(1,400,600,600)

would give a square of the same size in position (600,600) but drawn with dotted lines.

The procedure first calculates the correct 'PLOT' number for the type of line required, and assigns this value to the variable 't'. This is then used with the PLOT instruction to give dotted or solid lines. The facility to replace the 'PLOT' number with a variable is a very useful and flexible feature of the PLOT instruction. Some surprising results may often be had when a little ingenuity is used.

PLOTTING POINTS

Another useful variation on the PLOT instruction provides for the plotting of a point, rather than a line. Again the position of the point can either be specified as an absolute position on the screen (using PLOT69) or relative to the last position (using PLOT65). This enables single points to be positioned anywhere you want on the screen.

SUMMARY OF PLOT INSTRUCTIONS

All PLOT instructions have the same basic format of

PLOT k,x,y

where x,y is a relative or an absolute position on the screen, and k determines the plotting function carried out. Those that we have so far described are as follows:

Value of k	Function
0	Move relative to last point.
1	Draw a solid line relative.
4	Move to absolute position. (Same as MOVE x,y)
5	Draw a solid line absolute. (Same as DRAW x,y)
17	Draw a dotted line relative.
21	Draw a dotted line absolute.
65	Plot single point relative.
69	Plot single point absolute.
73	Fill left and right relative.
77	Fill left and right absolute.
81	Plot & fill triangle relative.
85	Plot & fill triangle absolute.

These are not, of course, the only PLOT instructions available, as a quick glance at the User Guide will soon show. They are, however, the ones most commonly used. Note how most of the instructions are in pairs, for relative and absolute plotting, and that in each pair, the 'PLOT' numbers differ by 4 exactly. Patterns like this can often be used to produce efficient and flexible programs.

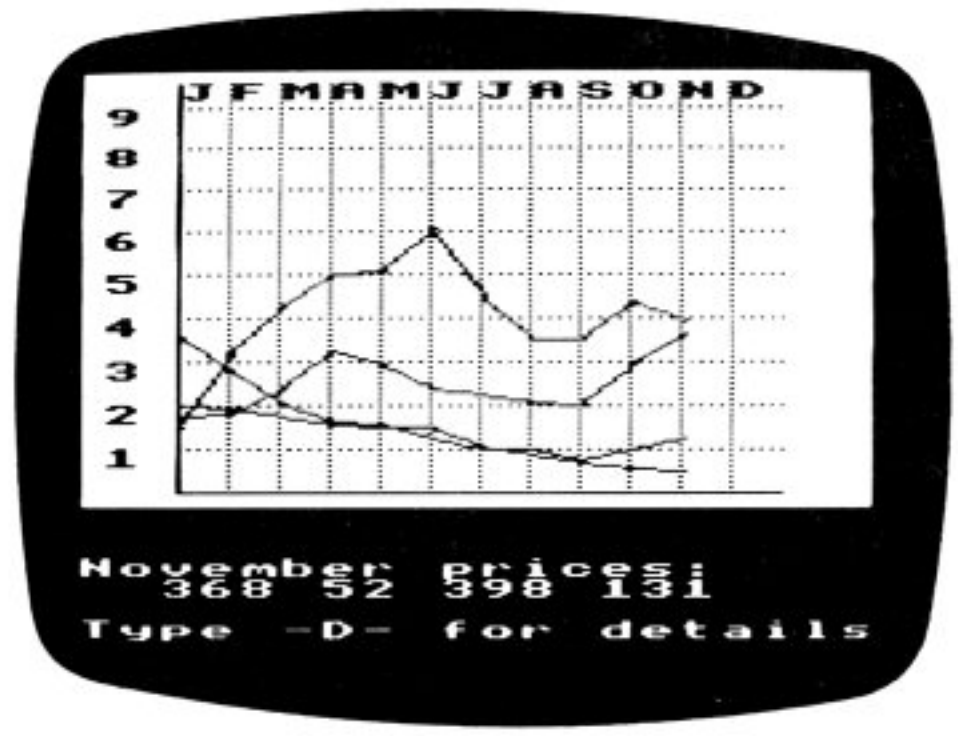
Next month, as already mentioned, we shall look at some of the interesting possibilities available when we start manipulating colour. One clever effect that we shall describe, is how to make one moving object appear to travel behind, or in front of, a stationary object.

PLAY THE STOCK MARKET

by Nick Day

Have you ever fancied your hand playing the stock market? Did you ever want the opportunity to gamble somebody else's money away on the stock exchange, and did you feel that the whole business was a little too complicated to get to grips with? Well we'll give you a chance with this program. It's the computer's money, which is a good start, it's confined to a limited market for simplicity, and it's fun.

The computer gives you (yes, gives you) a thousand pounds to start with. It's then up to you to let your entrepreneurial skills have full reign, and sensibly invest that windfall to make more. You're given a choice of four companies to put money into, those being fictitious firms with familiar sounding names like 'Red' or 'Blue'. Monthly reports are prepared, in graphic form, from which you plan your financial strategy, and the computer will then, under your instructions, carry out the buying and selling of shares as required.



When you run the pogram it will ask you for the number of players (or investors). This may be from just one up to six. Then it will begin its first cycle by displaying a graph showing the relative share prices of the four companies; a different colour for each. Before entering the transaction phase, where all the buying and selling is done, each player is given an opportunity to digest the graphic, and tabular information available to him. Separate tables of data are compiled on an individual basis. Important information about a player's financial situation and share costs are presented in an analytical format. Switching backwards and forwards between the two displays is easily done by pressing 'G' for the graph, or 'D', for the data table, until the player is quite satisfied he has a plan of action.

The transaction phase is entered from the data display by typing 'T', to sell off bought stocks or to buy more. The end of the transaction phase is marked by the pressing of the 'F' key, whereupon the program will return to a graph display and prompt for the next broker's instructions. It will prompt for all information needed from each player. When all the players have entered their transactions (and each must complete a minimum of three per month), the program will go into a quick calculation phase for that month. The graph returns to show the latest prices for Messrs. Red and Blue etc. Then you'll begin to get an idea about how well you've done. This cycle repeats itself once for every month during a year, and at the end of that year each broker's financial standing is displayed for assessment.

PROGRAM NOTES

Two main procedures, PROCdrawgraph and PROCtable, are responsible for drawing the graph and printing the results tables at the end of each month. PROCask, PROCtrans and PROCact deal with the transaction phase.

Two other procedures which could be useful in other applications are PROCwait and PROCprcol. The former is a delay loop that will wait for the number of seconds that are given as the parameter, 'W%', useful for those occasions, as here, when you may wish

to display a message for a certain period of time before changing it. PROCprcol prints numbers (the parameter 'no') in column form by printing the integer part in a specified field width, 'xp%'. The decimal part can be optionally printed (to print just the integer part, 'int' set to TRUE) but only to 2 places.

```
   10 REM Program STOCKS
   20 REM Version E1.2
   30 REM Author N.Day
   40 REM ELBUG June 1984
   50 REM Program subject to copyright
   60 :
  100 MODE1
  110 *FX4,2
  120 ON ERROR GOTO 360
  130 INPUTTAB(5,12)"How many brokers "
,play%
  135 VDU23,224,-1;-1;-1;-1,0
  140 PROCdim:PROCsetup:@%=&10
  150 REPEAT
  160 r%=0:g%=0:y%=0:b%=0
  170 month%=month%+1
  180 IF month%=1 THEN PROCjan
  190 IF month%=2 THEN PROCfebadj
  200 IF month%>2 THEN PROCadj
  210 FOR turn%=1 TO play%
  220 REPEAT
  230 MODE2:PROCdrawgraph:MODE1
  240 *FX11,0
  250 PROCtable(turn%):PROCgraph
  260 UNTIL g$="T"
  270 REPEAT
  280 PROCask
  290 UNTIL s$="F" AND trans>2
  300 PROCwait(2)
  310 NEXT
  320 UNTIL month%=12
  330 PROCend
  340 END
  350 :
  360 ON ERROR OFF:MODE 6
  370 *FX11,20
  380 *FX4,0
  390 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
  400 END
  410 :
 1000 DEF PROCjan
 1010 BP(1)=FNprice:RP(1)=FNprice
 1020 YP(1)=FNprice:GP(1)=FNprice
 1030 ENDPROC
 1040 :
 1050 DEF FNprice
 1060 h=RND(4)-1
 1070 =(h*100)+RND(100)+50
 1080 :
 1090 DEF PROCsetup
 1100 VDU23;8202;0;0;0;:month%=0
 1110 RS=0:GS=0:YS=0:BS=0:trans=0
 1120 ENDPROC
 1130 :
 1140 DEF PROCdim
 1150 PROCdimonths:VDU23;8202;0;0;0;
 1160 DIM BP(12):DIM YP(12)
 1170 DIM GP(12):DIM RP(12)
 1180 DIM cash(play%):DIM E(play%)
 1190 DIM RS(play%):DIM YS(play%)
 1200 DIM GS(play%):DIM BS(play%)
 1210 DIM tsv(play%)
 1220 FOR turn%=1 TO play%
 1230 cash(turn%)=100000
 1240 NEXT
 1250 DIM xdim(13):RESTORE 1290
 1260 FOR X%=1 TO 13
 1270 READ place:xdim(X%)=place
 1280 NEXT
 1290 DATA 160,240
 1300 DATA 320,400,480,560
 1310 DATA 640,720,800,880
 1320 DATA 960,1040,1120,1200
 1330 DATA 1280
 1340 ENDPROC
 1350 :
 1360 DEF PROCdrawgraph
 1370 VDU23;8202;0;0;0;
 1380 PROCwin:GCOL0,0:VDU5:
 1390 MOVE160,1000:PRINT"J"
 1400 MOVE160,1000:DRAW160,384
 1410 DRAW1120,384:RESTORE 1510
 1420 FOR X%=240 TO 1100 STEP 80
 1430 MOVE X%,1000:READm$:PRINTm$
 1440 MOVE X%,1000:PLOT21,X%,384
 1450 NEXT
 1460 P=0
 1470 FOR Y%=448 TO 960 STEP 64
 1480 P=P+1:MOVE 40,Y%:PRINT;P
 1490 MOVE 160,Y%:PLOT21,1120,Y%
 1500 NEXT
```

```
1510 DATA F,M,A
1520 DATA M,J,J,A
1530 DATA S,O,N,D
1540 VDU4:PROCplot(xdim(1),BP(1),6)
1550 PROCplot(xdim(1),RP(1),1)
1560 PROCplot(xdim(1),YP(1),3)
1570 PROCplot(xdim(1),GP(1),2)
1580 IF month%>1 THEN PROClines
1590 PROCprint
1600 IF month%=12 THEN PROCwait(3):END
PROC
1610 REPEAT a$=GET$:UNTIL a$="D"
1620 ENDPROC
1630 :
1640 DEF PROCplot(X%,P,C)
1650 GCOL0,C:Y%=(((P/100)*16)*4)+384
1660 PLOT69,X%+10,Y%
1670 ENDPROC
1680 :
1690 DEF PROClines
1700 FOR M%=2 TO month%
1710 PROCdraw(xdim(M%),BP(M%),BP(M%-1)
,6)
1720 PROCdraw(xdim(M%),RP(M%),RP(M%-1)
,1)
1730 PROCdraw(xdim(M%),YP(M%),YP(M%-1)
,3)
1740 PROCdraw(xdim(M%),GP(M%),GP(M%-1)
,2)
1750 NEXT
1760 ENDPROC
1770 :
1780 DEF PROCdraw(X%,P,L,C)
1790 GCOL0,C:Y%=(((L/100)*16)*4)+384
1800 MOVE xdim(M%-1),Y%
1810 Y%=(((P/100)*16)*4)+384
1820 PLOT5,X%+10,Y%
1830 ENDPROC
1840 :
1850 DEF PROCprint
1860 IF play%>1 THEN PRINTTAB(2,22)"BR
OKER NUMBER ";turn%'
1870 PRINTmonth$(month%);" prices:"
1880 COLOUR1:PRINTTAB(2,27);RP(month%);
1890 COLOUR2:PRINT" ";GP(month%);
1900 COLOUR3:PRINT" ";YP(month%);
1910 COLOUR6:PRINT" ";BP(month%)'
1920 IF month%=12 THEN ENDPROC
1930 COLOUR7:PRINTTAB(1,30)"Type -D- f
or details"
1940 ENDPROC
1950 :
1960 DEF PROCdimonths
1970 DIM month$(12):RESTORE 2010
1980 FOR M%=1 TO 12
1990 READ month$:month$(M%)=month$
2000 NEXT
2010 DATA January,February,March
2020 DATA April,May,June
2030 DATA July,August,September
2040 DATA October,November,December
2050 ENDPROC
2060 :
2070 DEF PROCtable(T%)
2080 VDU23;8202;0;0;0;
2090 VDU19,0,4,0,0,0:VDU19,3,2,0,0,0
2100 COLOUR0:COLOUR130
2110 PRINTTAB(1,1)STRING$(38,CHR$224)
2120 PRINTTAB(1,9)STRING$(38,CHR$224)
2130 PRINTTAB(4,13)STRING$(35,CHR$224)
2140 PRINTTAB(4,17)STRING$(35,CHR$224)
2150 PRINTTAB(4,5)STRING$(35,CHR$224)
2160 PRINTTAB(1,2)SPC(38)
2170 PRINTTAB(1,10)SPC(38)
2180 PRINTTAB(4,14)SPC(35)
2190 PRINTTAB(4,18)SPC(35)
2200 PRINTTAB(4,6)SPC(35)
2210 PRINTTAB(2,2);month$(month%);" Pr
ices (in p.) :"
2220 PRINTTAB(20,6)"BLUE";
2230 PROCprcol(7,BP(month%),TRUE)
2240 PRINTTAB(2,10);
2250 IF play%>1 THEN PRINT"Broker numb
er ";T%;" - ";
2260 PRINT"Your stocks :"
2270 COLOUR1
2280 PRINTTAB(5,6)"RED";
2290 PROCprcol(7,RP(month%),TRUE)
2300 COLOUR3:COLOUR128
2310 PRINTTAB(5,4)"GREEN";
2320 PROCprcol(5,GP(month%),TRUE)
2330 PRINTTAB(5,12);"GREEN";
2340 PROCprcol(6,GS(T%),TRUE)
2350 PRINT"  worth: £";
2360 PROCprcol(3,(GS(T%)*GP(month%))/1
00,FALSE)
2370 COLOUR2
2380 PRINTTAB(20,4)"YELLOW";
2390 PROCprcol(5,YP(month%),TRUE)
2400 COLOUR1:COLOUR130
2410 PRINTTAB(5,14);"RED";
2420 PROCprcol(8,RS(T%),TRUE)
2430 PRINT"  worth: £";
2440 PROCprcol(3,(RS(T%)*RP(month%))/1
00,FALSE)
2450 COLOUR2:COLOUR128
2460 PRINTTAB(5,16);"YELLOW";
2470 PROCprcol(5,YS(T%),TRUE)
2480 PRINT"  worth: £";
2490 PROCprcol(3,(YS(T%)*YP(month%))/1
00,FALSE)
2500 COLOUR0:COLOUR130
2510 PRINTTAB(5,18);"BLUE";
2520 PROCprcol(7,BS(T%),TRUE)
2530 PRINT"  worth: £";
2540 PROCprcol(3,(BS(T%)*BP(month%))/1
00,FALSE)
2550 COLOUR2:COLOUR128
```

```
 2560 tsv(T%)=((GS(T%)*GP(month%))+(YS(
T%)*YP(month%))+(BS(T%)*BP(month%))+(RS
(T%)*RP(month%)))/100
 2570  PRINTTAB(4,22)"Total stock value
 :  £";
 2580 PROCprcol(5,tsv(T%),FALSE)
 2590 PRINTTAB(4,23)"Cash reserves
:  £";
 2600 PROCprcol(5,cash(T%)/100,FALSE)
 2610 PRINTTAB(4,24)"You are worth
:  £";
 2620 PROCprcol(5,tsv(T%)+(cash(T%)/100
),FALSE)
 2630 ENDPROC
 2640 :
 2650 DEF PROCgraph
 2660 PRINTTAB(1,27)"Type -G- to see gr
aph again"'"   or -T- to commence trans
actions"
 2670 move=0:g$=GET$
 2680 ENDPROC
 2690 :
 2700 DEF PROCask
 2710 trans=r%+g%+y%+b%
 2720 PRINTTAB(1,27)"Which share would
you like to deal in?"
 2730 PRINTTAB(1,28)"-R / G / Y / B- or
 -F- for finished   "
 2740 s$=GET$
 2750 IF s$="R" THEN PROCtrans("RED",RP
(month%),RS(turn%),cash(turn%)):cash(tu
rn%)=R%:RS(turn%)=RS(turn%)+change:IF r
%=0 AND change<>0 THEN r%=1
 2760 IF s$="G" THEN PROCtrans("GREEN",
GP(month%),GS(turn%),cash(turn%)):cash(
turn%)=R%:GS(turn%)=GS(turn%)+change:IF
 g%=0 AND change<>0 THEN g%=1
 2770 IF s$="Y" THEN PROCtrans("YELLOW"
,YP(month%),YS(turn%),cash(turn%)):cash
(turn%)=R%:YS(turn%)=YS(turn%)+change:I
F y%=0 AND change<>0 THEN y%=1
 2780 IF s$="B" THEN PROCtrans("BLUE",B
P(month%),BS(turn%),cash(turn%)):cash(t
urn%)=R%:BS(turn%)=BS(turn%)+change:IF
b%=0 AND change<>0 THEN b%=1
 2790 PROCtable(turn%)
 2800 IF s$="F" AND trans<3 THEN PROCwa
rntrans
 2810 PRINTTAB(1,27);SPC(38)TAB(1,28);S
PC(38)
 2820 ENDPROC
 2830 :
 2840 DEF PROCtrans(c$,P%,S%,C%)
 2850 REPEAT:PROCact:R%=C%-(change*P%):
IF R%<1 THEN PROCwarn("CASH")
 2860 UNTIL R%>0
 2870 ENDPROC
 2880 :
 2890 DEF PROCwarntrans
 2900 COLOUR0:COLOUR130
 2910 PRINTTAB(1,27)"  You must make t
hree transactions!   "TAB(1,28)SPC(38)

 2920 VDU7:PROCwait(3)
 2930 COLOUR2:COLOUR128
 2940 ENDPROC
 2950 :
 2960 DEF PROCwin:VDU24,10;364;1270;1020
;:VDU28,0,31,19,23:GCOL0,135:CLG:ENDPROC
 2970 :
 2980 DEF PROCact
 2990 REPEAT
 3000 G%=TRUE:IF S%=0 THEN t$="B"
 3010 IF S%<>0 THEN PRINTTAB(1,27)SPC(3
8):PRINTTAB(1,27)"Are you buying or sel
ling in "c$;"?":PRINTTAB(1,28)SPC(38):P
RINTTAB(1,28)SPC(18);"(B/S)":t$=GET$
 3020 IF t$="B" THEN tran$="buy"
 3030 IF t$="S" THEN tran$="sell"
 3040 PRINTTAB(1,27)SPC(38):PRINTTAB(1,
27)"INPUT the number of ";c$;" shares":
PRINTTAB(1,28)SPC(38):PRINTTAB(1,28)"
 you wish to ";tran$;:INPUTchange
 3050 IF change<0 THEN PRINTTAB(30,28)"
SILLY!":PROCwait(2):G%=FALSE
 3060 IF t$="S" AND change>S% THEN PROC
warn("SHARES"):G%=FALSE
 3070 IF t$="B" AND (change*P%)>C% THEN
 PROCwarn("cash"):G%=FALSE
 3080 UNTIL G%=TRUE
 3090 IF t$="S" THEN change=-change
 3100 ENDPROC
 3110 :
 3120 DEF PROCwarn(W$)
 3130 COLOUR0:COLOUR130
 3140 VDU7:PRINTTAB(1,27)SPC(9);"Insuff
icient ";W$;SPC(12):PRINTTAB(1,28)SPC(3
8):PROCwait(3)
 3150 COLOUR2:COLOUR128
 3160 ENDPROC
 3170 :
 3180 DEF PROCfebadj
 3190 RP(month%)=RP(month%-1)+FNadj1(RP
(month%-1))
 3200 GP(month%)=GP(month%-1)+FNadj1(GP
(month%-1))
 3210 YP(month%)=YP(month%-1)+FNadj1(YP
(month%-1))
 3220 BP(month%)=BP(month%-1)+FNadj1(BP
(month%-1))
 3230 ENDPROC
 3240 :
 3250 DEF PROCadj
 3260 RP(month%)=RP(month%-1)+FNadj2(RP
(month%-1),RP(month%-2))
 3270 GP(month%)=GP(month%-1)+FNadj2(GP
(month%-1),GP(month%-2))
 3280 YP(month%)=YP(month%-1)+FNadj2(YP
(month%-1),YP(month%-2))
```

→

```
 3290 BP(month%)=BP(month%-1)+FNadj2(BP
(month%-1),BP(month%-2))
 3300 ENDPROC
 3310 :
 3320 DEF FNadj1(P%)
 3330 R=RND(1):F%=RND(3)+1
 3340 IF P%<5 THEN P%=50
 3350 Q%=RND(P%/F%)
 3360 IF R<.5 THEN Q%=-Q%
 3370 IF R<.2 THEN Q%=RND(200)
 3380 IF Q%<-P%+1 THEN Q%=RND(50)
 3390 IF P%+Q%>900 THEN Q%=Q%-RND((P5+Q
%)-900)
 3400 =Q%
 3410 :
 3420 DEF FNadj2(P%,L%)
 3430 N%=P%
 3440 R=RND(1):F%=RND(3)+1
 3450 IF P%<100 THEN P%=100:F%=RND(2)
 3460 Q%=RND(P%/F%)
 3470 IF R<.05 THEN Q%=RND(200)
 3480 IF  R<.2 THEN Q%=-Q%
 3490 IF N%<L% THEN Q%=-Q%
 3500 IF Q%<-N%+1 THEN 3460
 3510 IF P%+Q%>890 THEN 3440
 3520 =Q%
 3530 :
 3540 DEF PROCend
 3550 FOR turn%=1 TO play%
 3560 E(turn%)=(cash(turn%)-100000)/100
 3570 a$="made":b$=" "
 3580 IF E(turn%)>1000 THEN b$="Well do
ne!"
 3590 IF cash(turn%)<100000 THEN a$="lo
st":E(turn%)=(100000-cash(turn%))/100:b
$="Bankrupt!"
 3600 CLS:CLG:@%=&20206
 3610 PRINTTAB(6,7)"Broker number";:PRO
Cprcol(2,turn%,TRUE)
 3615 PRINT", you have ";a$
 3616 PRINTTAB(14,9)" £";E(turn%);". ";
b$
 3617 PROCwait(10)
 3620 NEXT
 3630 ENDPROC
 3640 :
 3650 DEF PROCwait(W%):now=TIME:REPEAT:U
NTILTIME=now+(W%*100):ENDPROC
 3660 :
 3670 DEF PROCprcol(xp%,no,int)
 3680 PRINTSPC(xp%-LEN(STR$(INT(no)))+1
)STR$(INT(no));
 3690 IF int THEN ENDPROC
 3700 dec=no-INT(no):t%=INT(10*dec)
 3710 u%=INT((dec*10-t%)*10)
 3720 PRINT".";LEFT$(STR$(t%),1);LEFT$(
STR$(u%),1)
 3730 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

FUNCTION KEY TO LIST A PROGRAM IN PAGED MODE - R.M.Denby

A function key may be given a short definition that will list a program in paged mode automatically every time Func-f0 is pressed:

*K.0|NL.|M <Return>

(The '|' character can be obtained by pressing Shift and the right cursor key).

(Explanation: *K.0 - *KEY 0
|N - CNTRL N, for paged mode
L. - LIST, abbreviated
|M - CNTRL M, end of function key definition).

CASSETTE LOADING TIMES - C.Lyne

The approximate loading time from cassette can be given in seconds by the statement PRINT &####/&4E, where #### is the program length printed on the screen after a sample load.

ON ERROR OFF - P.Mabey

Peter Mabey suggests that readers be reminded that when writing an error trap routine, the first statement in the routine should turn off error trapping. Otherwise any mistake therein will make it loop back to the beginning, so that the only way to stop it is to Break. ie.:

```
  10 ON ERROR GOTO 900
  20 REM the rest of the program
 900 ON ERROR OFF
 910 REM error handling routine
```

This is always incorporated into ELBUG programs and for a generally useful error handling routine, see how one of the published programs is written.

SCREEN FREEZER

by David Graham and Michael Cope

Have you ever wished you could take a pause in the middle of your favourite arcade game - either because you have been called away to the telephone in the middle of a high-scoring run, or simply so that you can examine the screen in peace while you work out a strategy for the next phase of the game. FREEZER allows you to do just that.

FREEZER is a short machine code routine that is loaded before a game or other program to provide a pause control. It works perfectly well on most of the arcade games that we have used it with - see table - and it works equally well with Basic or machine code programs.

Games successfully frozen:

Invaders	Superior Software
Centibug	Superior Software
Fruit Machine	Superior Software
Swoop	Program Power
Croaker	Program Power

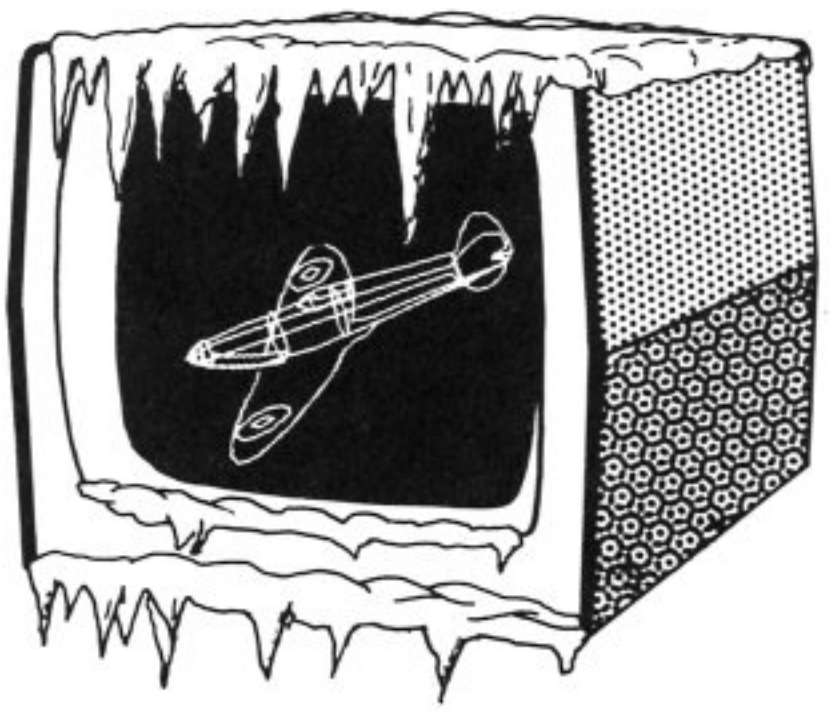
Note that if one of the above games refuses to freeze, it may be that you have a different version of the game to us. Monsters by Acornsoft refused to be frozen.

Once FREEZER is activated, pressing the '-' (minus) key during the playing of most games will halt the program. Pressing it a second time will resume the game. If your game uses any sound effects then these will also be 'frozen'. The delay button may be pressed again to produce further delays.

Pressing Break disables the routine, but it may be re-enabled by typing CALL &A00 <return>. A beep indicates that it has been successfully called.

DETAILED INSTRUCTIONS

Type in the program, and save it on cassette. The main routine is in assembler and this is converted to machine code which must be stored in a safe area of memory, where it will not be affected by the game program. When you run the program you will be asked



for the address of the memory area to be used. This may be in hex or decimal, but hex values should be preceded by '&'. Generally speaking you should enter &A00 here, unless you wish to locate the code elsewhere (if you suspect that your arcade game uses this particular memory space). You should then see the assembly listing scroll on the screen, and a beep should be heard. The machine should now behave exactly as normal, until you press the '-' key. It should then freeze completely until you press the '-' key again, when everything will continue as before.

Once the routine is working you should save a copy of the machine code by typing:
*SAVE FREEZER A00 A40 <return>
You can now reload the code at any time by using *RUN FREEZER - again a beep will indicate successful execution.

Once the code is in your machine, and enabled, you can load and run any game you wish. Remember though, to execute CALL &A00 after pressing Break at any time.

CHANGING THE FREEZER KEY

To change the key used to control the freeze action, alter the value in line 130 to equal the positive value of the negative INKEY value plus 1 (see User Guide page 159). For example, to use key 'F', find the negative INKEY value of 'F', which is -68, add 1 to this giving -67 (not -69), and dropping the sign results in the value of 67. This value should then be included at line 130 in place of the 23 listed. This rather complicated process is necessary as the internal representation of a key differs from the value usually seen in a Basic program. The negative INKEY values are all on page 159 of the User Guide.

```
10 REM PROGRAM FREEZE
20 REM AUTHORS D.GRAHAM / M.COPE
30 REM VERSION E0.1
40 REM ELBUG  JUNE 1984
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
100 ON ERROR GOTO 420
110 MODE 6
120 code%=&A00
130 keycode=23
140 FOR pass=0 TO 3 STEP 3
150 P%=code%
160 [OPT pass
170 .init
180 LDA#7:JSR&FFEE
190 LDA#14:LDX#2:JSR&FFF4
200 LDA#entry MOD 256:STA&220
210 LDA#entry DIV 256:STA&221
220 RTS
230 .entry
240 PHP:PHA:TXA:PHA:TYA:PHA
250 JSR key:BNE exit
260 .held
270 JSR key:BEQ held
280 .wait
290 JSR key:BNE wait
300 .held2
310 JSR key:BEQ held2
320 .exit
330 PLA:TAY:PLA:TAX:PLA:PLP:RTS
340 .key
350 LDA#122:JSR&FFF4
360 CPX#keycode:RTS
370 ]
380 NEXT
390 CALL init
400 END
410 :
420 ON ERROR OFF
430 MODE 6
440 IF ERR<>17 REPORT:PRINT" at line
";ERL
450 END
```

POINTS ARISING

SOUND ENVELOPE EDITOR

It has emerged that there was a potential error in the working of the Sound Envelope Editor published in issue 4 of ELBUG. This occured if a value was selected for the parameter 's' in excess of 83. Changing lines 360 (1 changed to 2) and 650 (4 changed to 5) as shown below will solve this problem.

```
360 DEFPROCedit(S%,F%):J%=0:FORI%=S%TO F%:NEXTF%:COLOUR3:J%=J%+1:PRINTTAB(J%*5,
20);P$(I%);:COLOUR1:PRINTTAB(J%*5-2,22);FNstr(E%(I%)):NEXT
650 DEFFNstr(n)=STRING$(5-(LEN(STR$(n))),CHR$32)+STR$(n)
```

DOMINOES

Because the original program had been partly compacted to save space before listing in the magazine, two of the lines will generate a "No such variable..." error if typed in as listed. The solution is to insert extra spaces as shown below (compare with the original):

```
 330 IF ELK% PROCELK ELSE PROCdisphand(1,0):PROCplayer
1470 IF bd%>pd% ELK%=TRUE ELSE ELK%=FALSE:s$=n$
```

The absence of crucial spaces causes Basic to fail to recognise keywords like ELSE and THEN. We shall be publishing a series of articles starting in next month's issue, which will examine problems such as these in more detail.

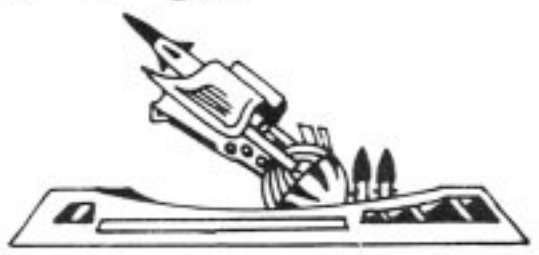
GALACTIC INVASION

by L.P. Baxter

This program provides an impressive demonstration of how good arcade games can be when written in Basic on the Electron. The movement of the graphics appears quite smooth, and the presentation is very professional.

The principle of this game is for you to shoot down the Galactic Invaders which appear as a massed force on the screen. As you might expect, they put up a good fight, swooping down one by one to attack you with purposeful but unpredictable trajectories. You move left and right across the bottom of the screen using the 'Z' and 'X' keys and fire back using the '.' key. When only one invader remains, a new and even faster attack strategy is engaged.

The different invaders rate different scores if you destroy them, as shown in the table at the start of the game. The highest scoring invaders vary randomly in value. If you successfully defend yourself against the first attack, you then face a second and further screen of invaders. You have three lives before your defences are exhausted. At any time during the display of the scoring or high score table, pressing the space bar will start the next game. Like many good games programs this one is quite long and you must type it in carefully to avoid mistakes. Pay particular attention to the spaces in the program, though if you do get them wrong, it is only the initial displays which are likely to be affected. The mass of data statements at the end of the listing establishes the invaders' swoop patterns, so are not as critical as they might be, but errors will be thrown up if you do not have enough data. Please note that this program will not work if it is renumbered because of the use of variable RESTORE statements.

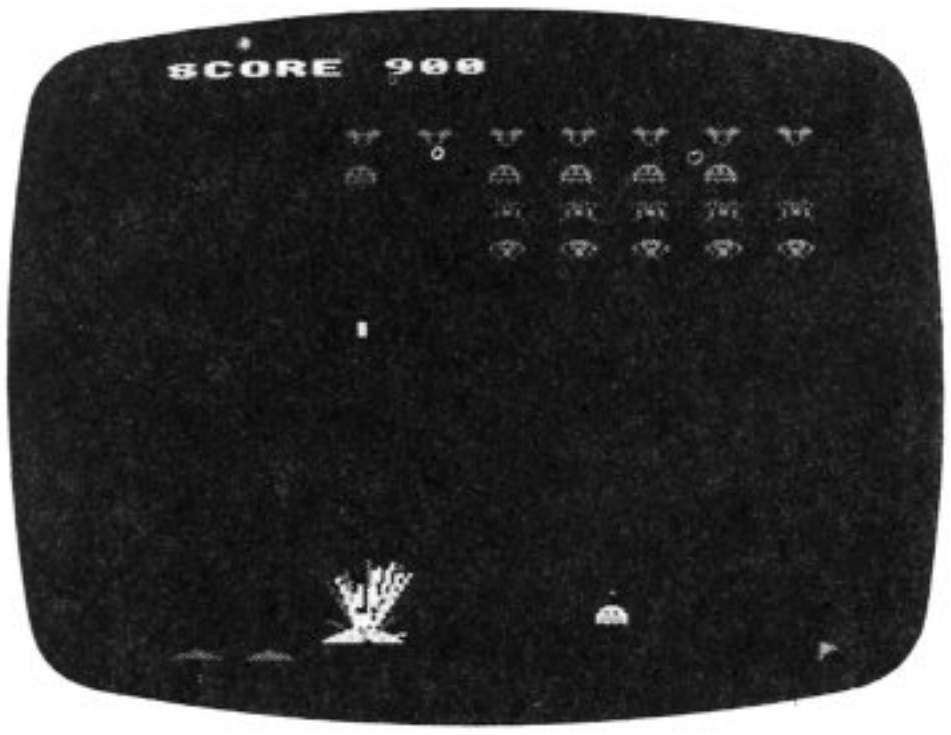


```
  10 REM Program GALACTIC INVASION
  20 REM Verson E1.0
  30 REM Author L.P.Baxter
  40 REM ELBUG JUNE 1984
  50 REM Program subject to Copyright
  60 :
 100 ON ERROR GOTO 20170
 110 PROCe
 120 DIM A$(4),B$(4),X%(10),Y%(10),N$(
5),S%(5)
 130 FOR G%=1 TO 5:S%(G%)=6000-G%*1000
:N$(G%)="ELBUG":NEXT
 140 REM==== COLD START ====
 150 MODE5:FORA%=1 TO 4:B$(A%)=CHR$(22
4+A%):NEXT
 160 VDU19,0,6,0,0,0:VDU19,3,4,0,0,0
 170 VDU19,2,5,0,0,0:PROC1
 180 P%=0:R%=0
 190 E$=CHR$230:C$=CHR$231:D$=CHR$238:
F$=CHR$235+CHR$236
 200 B$=CHR$32+CHR$232+CHR$233+CHR$234
+CHR$32:J%=0:T%=2
 210 n%=1:G$=C$:N%=30
 220 U%=3:K%=8:H$=CHR$229
 230 REM==== WARM START ====
 240 W%=0:R%=0:Z%=0:M%=0
 250 VDU 23,1,0;0;0;0;
 260 CLS:COLOUR3:PRINT"SCORE":F%=0
 270 V%=28:REM  FLAG FOR LAST MAN(NO.
INV LEFT)
 280 FORA%=1 TO 4:A$(A%)=STRING$(6,CHR
$(224+A%)+CHR$32)+CHR$(224+A%):NEXT
 290 H%=0:D%=0:E%=0:I$="":VP=0:HP=0:I%
=2
 300 REPEAT PROCd:UNTIL A%
 310 PRINTTAB(0,31);STRING$(T%,F$);TAB
(19-LEN(G$));G$;
 320 L%=0:PROCa:y%=0:x%=0
 330 :
 340 REPEAT
 350 IF INKEY-104 AND Z%=0 PROCb
 360 IF INKEY-98 L%=-1:PROCa ELSE IF I
NKEY-67 L%=1:PROCa
 370 PROCc:PROCh:PROCc
 380 COLOUR2:PRINT TAB(6,0);W%+P%
 390 IF W%=7000 THEN PROCk:y%=-1:GOTO
470
 400 IF F% PROCi
 410 IF T%=-1 PROCj:x%=-1:GOTO 470
 420 REM NOW INVADERS TURN
 430 IF H%=0 AND RND(20)=1 THEN PROCd
 440 IF H% PROCf:GOTO 460
 450 IF RND(N%)=1 OR V%=1 THEN PROCg
 460 PROCc
 470 UNTIL y% OR x%
 480 IF y% THEN 240 ELSE 150
 490 END
 500 :
 510 DEF PROCa
 520 IF J%+L%<0 OR J%+L%>15 THEN ENDPR
OC
 530 J%=J%+L%
 540 COLOUR2:PRINT TAB(J%,30);B$;
 550 ENDPROC
 560 :
 570 DEF PROCb
 580 COLOUR3:Z%=J%+2:M%=29
 590 PRINTTAB(Z%,M%);H$
 600 ENDPROC
 610 :
 620 DEF PROCc
 630 IF Z%=0 ENDPROC
 640 IF M%=4 THEN PRINT TAB(Z%,M%);SPC
1:Z%=0:M%=0:ENDPROC
 650 M%=M%-1:IF ?(Z%*16+M%*320+2+HIMEM
) THEN 670
 660 COLOUR2:PRINTTAB(Z%,M%+1);SPC1;CH
R$11;CHR$8;H$:ENDPROC
 670 PRINTTAB(Z%,M%+1);SPC1;CHR$11;CHR
$8;"*";CHR$8;
 680 IF NOT(D%=POS AND E%=VPOS) THEN 7
30
 690 H%=0:A$(VP)=LEFT$(A$(VP),HP*2-2)+
CHR$32+MID$(A$(VP),HP*2)
 700 W%=W%+500-100*VP:VDU 32:IF VP<>1
THEN P%=P%+500-100*VP:GOTO 760
 710 S%=RND(10)*100:P%=P%+S%:PRINT CHR
$8;CHR$8;S%+400;
 720 PROCr:VDU8,8,8,8,32,32,32,32:GOTO
770
 730 W%=W%+500-FNY(VPOS)*100
 740 A%=FNY(VPOS):B%=FNX(POS)*2-1:A$(A
%)=LEFT$(A$(A%),B%-1)+CHR$32+MID$(A$(A%
),B%+1)
 750 VDU 32:IF H%=1 THEN PROCf
 760 PROCq
 770 Z%=0:M%=0:V%=V%-1
 780 IF V%=1 THEN N%=2:U%=0
 790 ENDPROC
 800 :
 810 DEF PROCd
 820 COLOUR1
 830 A%=RND(3)-2:IF A%=0 THEN ENDPROC
 840 IF I%+A%<1 OR I%+A%>4 THEN I%=I%-
A% ELSE I%=I%+A%
 850 PRINT TAB(I%,4);SPC1;A$(1);SPC1''
TAB(I%);SPC1;A$(2);SPC1''TAB(I%);SPC1;A
$(3);SPC1''TAB(I%);SPC1;A$(4);SPC1:ENDP
ROC
 860 :
 870 DEF PROCe
 880 VDU 23,225,66,231,255,102,36,36,3
6,24
 890 VDU 23,226,56,124,84,214,254,130,
254,170
 900 VDU 23,227,102,153,36,90,90,90,66
,129
 910 VDU 23,228,60,126,165,189,153,90,
24,36
```

```
  920 VDU 23,229,0,24,24,24,24,24,24,0
  930 VDU 23,230,0,8,8,8,8,8,8,0
  940 VDU 23,231,48,56,60,56,48,32,32,32
  950 VDU 23,232,0,0,0,1,3,7,15,31
  960 VDU 23,233,24,60,60,255,255,255,2
55,255
  970 VDU 23,234,0,0,0,128,192,224,240,
248
  980 VDU 23,235,0,0,0,1,3,15,31,63
  990 VDU 23,236,0,0,0,0,128,224,240,248
 1000 VDU 23,237,0,0,0,255,0,255,0,0,0
 1010 VDU 23,238,176,184,188,184,176,16
0,160,160
 1020 ENDPROC
 1030 :
 1040 DEF PROCf
 1050 IF V%=1 THEN SOUND 17,-15,30,1:SO
UND 17,-15,10,1:GOTO 1070
 1060 SOUND &0011,-15,r%,5:r%=r%-1
 1070 IF D%<0 OR D%>19 THEN 1090
 1080 PRINT TAB(D%,E%);SPC1;:IF R%<U% A
ND RND(K%)=1 AND E%<28 THEN R%=R%+1:X%(
R%)=D%:Y%(R%)=E%
 1090 IF E%>29 THEN 1120
 1100 READ A%,B%:D%=D%+A%:E%=E%+B%:IF D
%>=0 AND D%<20 THEN PRINT TAB(D%,E%);I$;
 1110 ENDPROC
 1120 IF V%=1 THEN 1130 ELSE IF RND(5)=
1 THEN PROCd ELSE PRINT TAB(FNX1(HP),FN
Y1(VP));I$
 1130 R%=0:H%=0
 1140 IF D%>J% AND D%<J%+4 THEN F%=1
 1150 ENDPROC
 1160 :
 1170 DEF PROCg
 1180 RESTORE (RND(3)-1)*20+11000
 1190 VP=RND(4):IF INSTR(A$(VP),B$(VP))
=0 THEN 1190
 1200 IF RND(2)=1 THEN 1220 ELSE A%=15
 1210 A%=A%-2:IF MID$(A$(VP),A%,1)=CHR$
32THEN 1210 ELSE HP=(A%+1)/2:GOTO 1240
 1220 HP=(INSTR(A$(VP),B$(VP))+1)/2
 1230 RESTORE (RND(3)-1)*20+10000
 1240 I$=B$(VP)
 1250 IF V%=1 THEN RESTORE (10060+(RND(
2)-1)*1000):PRINT TAB(FNX1(HP),FNY1(VP)
);SPC1;:HP=3
 1260 D%=FNX1(HP):E%=FNY1(VP)
 1270 r%=200:H%=1:ENDPROC
 1280 :
 1290 DEF FNX(X)=(X-I%+1)/2
 1300 DEF FNY(Y)=(Y-2)/2
 1310 DEF FNX1(X)=X*2+I%-1
 1320 DEF FNY1(Y)=2+Y*2
 1330 :
 1340 DEF PROCh
 1350 IFR%=0 THEN ENDPROC
 1360 FORA%=1TOR%:IF X%(A%)=0 AND Y%(A%
)=0 THEN NEXT:ENDPROC
 1370 PRINTTAB(X%(A%),Y%(A%));SPC1;:Y%(
A%)=Y%(A%)+1:IF Y%(A%)<30 THEN 1380 ELS
E C%=X%(A%):X%(A%)=0:Y%(A%)=0:IF C%<J%+
1 OR C%>J%+3 THEN 1390 ELSE F%=1:GOTO 1
390
 1380 COLOUR1:PRINTTAB(X%(A%),Y%(A%));E
$;
 1390 NEXT:PROCc:ENDPROC
 1400 :
 1410 DEF PROCi
 1420 T%=T%-1
 1430 COLOUR2:PRINT TAB(J%,30);B$;
 1440 PROCp:IF T%=-1 ENDPROC
 1450 CLS:COLOUR3:PRINT"SCORE"
```


```
 1460 COLOUR2:PRINT TAB(0,31);STRING$(T
%,F$);SPC2;TAB(19-LEN(G$));G$;TAB(0,30)
;B$
 1470 F%=0:H%=0:D%=0:E%=0:R%=0:HP=0:VP=
0:J%=0:M%=0:Z%=0
 1480 REPEAT:PROCd:UNTIL A%:ENDPROC
 1490 :
 1500 DEF PROCj
 1501 COLOUR2:COLOUR129:PRINTTAB(5,12)"
Game Over"
 1510 FOR A=1 TO 7000:NEXT
 1520 W%=W%+P%:IF W%<=S%(5) THEN 1690
 1530 REM MADE IT INTO HIGH SCORE TABLE
 1540 FOR A%=5 TO 1 STEP -1
 1550 IF S%(A%)<W% THEN C%=A%
 1560 NEXT:VDU22,6,19,0,1;0;19,1,3;0;
 1570 COLOUR129:COLOUR0:PRINTTAB(11,1)"
Galactic Invasion":COLOUR128:COLOUR1
 1580 PRINT'"Your score of ";W%;" is en
ough to rank"
 1590 PRINT TAB(0,4);C%;MID$("stndrdtht
h",C%*2-1,2);" in the high score table."
 1600 PRINT TAB(0,8);"Please enter your
 name in not more than ten letters."
 1610 PRINT TAB(14,15);">";SPC(10);"<";
SPC(13)
 1620 PRINT TAB(15,15);:*FX15,1
 1630 INPUT ""A$
```

```
 1640 IF LEN(A$)>10 THEN 1610
 1650 IF C%=5 THEN 1680
 1660 FOR A%=4 TO C% STEP -1
 1670 S%(A%+1)=S%(A%):N$(A%+1)=N$(A%):N
EXT
 1680 S%(C%)=W%:N$(C%)=A$
 1690 ENDPROC
 1700 :
 1710 DEF PROCk
 1720 P%=P%+W%:n%=n%+1
 1730 G$=STRING$(n% MOD 10,C$)+STRING$(
n% DIV 10,D$)
 1740 U%=3+INT(n%/2):IF U%>8 THEN U%=8
 1750 N%=N%-5:IF N%<5 THEN N%=2
```




```
1760 K%=11-U%
1770 FOR A=1 TO 2500:NEXT:ENDPROC
1780 :
1790 REM DISPLAY SHEETS
1800 DEF PROC1
1810 REPEAT:VDU 23,1,0;0;0;0;:*FX15,1
1820 Q%=0:PROCn:IF Q%=0 THEN ENDPROC
1830 CLS:C%=20:*FX15,1
1840 A$=" GALACTIC INVASION":PROCm(1)
 1850 A$=" "+STRING$(17,CHR$237):PROCm(
2)
 1860 A$="400  "+B$(1)+"   ?":PROCm(8)
 1870 A$="300  "+B$(2)+"  600":PROCm(12)
 1880 A$="200  "+B$(3)+"  400":PROCm(16)
 1890 A$="100  "+B$(4)+"  200":PROCm(20)
 1900 A$="Z KEY  LEFT":PROCm(26)
 1910 A$="X KEY  RIGHT":PROCm(28)
 1920 A$=". KEY  FIRE":PROCm(30)
 1930 IF INKEY$(1000)<>""THEN ENDPROC
 1940 UNTIL FALSE
 1950 :
 1960 DEF PROCm(A%)
 1970 IF A%=1 OR A%=2 GOTO1990
 1980 A$=STRING$(4,CHR$32)+A$
 1990 FOR B%=15 TO 1 STEP -1:A=SIN(97.6
):B$=MID$(A$,B%):PRINT TAB(0,A%);B$;:NE
XT:ENDPROC
 2000 :
 2010 DEF PROCn
 2020 CLS:A%=6
 2030 A%=A%-1
 2040 PRINT TAB(0,0);A%;". ";S%(A%);TAB
(10,0);N$(A%);CHR$30;
 2050 B%=0
 2060 B%=B%+1
 2070 VDU 11:IF INKEY10<>-1 THEN A%=0:B
%=5:ENDPROC
 2080 IF B%<4 THEN 2060
 2090 IF A%>1 THEN 2030
 2100 VDU 11,11,11:PRINT TAB(3,2)STRING
$(14,CHR$237);CHR$11;TAB(3,1)"HIGHEST S
CORES"
 2110 IF INKEY(1000)<>-1 THEN ENDPROC
 2120 Q%=1:ENDPROC
 2130 :
 2140 REM  EXPLOSION
 2150 DEF PROCp
 2160 FOR A%=1 TO 50
 2170 GCOL RND(4)-1,RND(3)
 2180 SOUND 1,0,RND(50)+105,1:SOUND 0,-
15,7,1
 2190 MOVE (J%+3)*64-32,32
 2200 DRAW RND(144)-96+(J%+3)*64,RND(14
4)+32
 2210 NEXT
 2220 FOR A=1 TO 3000:NEXT:ENDPROC
 9999 REM DATA FOR DIVES
10000 DATA -1,1,-1,1,-1,1,0,1,0,1,1,1,1
,1,1,1,1,1,1,1,1,0,1,0,1,1,1,0,1,0,1,1,
1,0,1,1,1,0,1,1,1,0,1,1,0,1,0,1,-1,1,-1
,0,-1,1,-1,0,-1,1,-1,1,-1,1,-1,1,-1,0,-
1,1,-1,1,-1,1,-1,1,-1,1
10020 DATA -1,1,-1,1,-1,1,0,1,0,1,1,1,1
,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,1,
1,0,1,0,1,0,1,1,1,0,1,1,0,1,-1,1,-1,0,-
1,0,-1,1,-1,0,-1,0,-1,1,-1,0,-1,0,-1,0,
-1,0,-1,1,0,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1
,0,1,1,1,1,1,1,1,1,1,1
10040 DATA -1,1,-1,1,-1,1,0,1,0,1,1,1,1
,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,1,
1,0,1,0,1,0,1,1,1,0,1,1,0,1,-1,1,-1,0,-
1,0,-1,1,-1,0,-1,0,-1,1,-1,0,-1,0,-1,0,
-1,0,-1,1, 0,1,1,1,-1,1,-1,1,-1,1,-1,1,
-1,1,0,1,1,1,1,1
10060 DATA 0,1,0,1,0,1,0,1,0,1,0,1,0,1,
0,1,0,1,0,1,-1,1,-1,1,-1,0,-1,1,1,1,1,1,0
, 1,1,1,0,1,0,1,0,1,0,1,1,1,1,-1,1,-1,1
,-1,0,-1,0,-1,0,-1,-1,-1,-1,0,-1,0,0,1,
-1,1,-1,1,0,0,0,1,0,0,0,1,0,1,1,1,1,1,0,1
,0,0,-1,1,-1,1
10070 DATA-1,1,-1,1,-1,1,-1,1,-1,1,-1,1
,-1,1,-1,1,-1,1
11000 DATA 1,1,1,1,1,1,1,0,1,0,1,-1,1,-1,
1,-1,1,-1,1,-1,1,-1,1,-1,0,-1,0,-1,1,-1,0,-1
,0,-1,1,-1,0,-1,1,-1,0,-1,1,-1,0,-1,1,0
,1,0,1,1,1,1,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,
1,0,1,1,1,1,1,1,1,1,1,1,1
```

```
11020 DATA 1,1,1,1,1,1,0,1,0,1,-1,1,-1,
1,-1,1,-1,1,-1,1,-1,0,-1,0,-1,0,-1,0,-1
,0,-1,1,-1,0,-1,0,-1,0,-1,1,-1,0,-1,1,0
,1,1,1,1,0,1,0,1,1,1,1,0,1,0,1,1,1,0,1,0,
1,0,1,0,1,1,0,1,-1,1,-1,1,-1,1,-1,1,-1,
0,-1,1,-1,0,-1,1,-1,1,-1,1,-1,1
11040 DATA 1,1,1,1,1,1,1,0,1,0,1,-1,1,-1,
1,-1,1,-1,1,-1,1,-1,0,-1,0,-1,0,-1,0,-1
,0,-1,1,-1,0,-1,0,-1,0,-1,1,-1,0,-1,1,0
,1,1,1,1,1,0,1,0,1,1,1,1,0,1,0,1,1,1,1,0,1,0,
1,0,1,0,1,1,0,1,-1,1,1,1,1,1,1,1,1,1,1,1,
1,0,1,-1,1,-1,1
11060 DATA 0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,
0,1,0,1,0,1,1,1,1,1,1,1,1,0,1,1,-1,1,-1,0,-
1,1,-1,0,-1,0,-1,0,-1,0,-1,1,-1,1,1,1,1,1
,1,1,0,1,0,1,0,1,-1,1,-1,0,-1,0,0,-1,-1
,-1,-1,-1,0,0,0,-1,0,0,0,-1,0,-1,1,-1,1
,0,1,0,0,1,1,1,1
11070 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1
20000 DEF PROCq
20010 A%=187:B%=179
20020 SOUND 1,-15,109,2
20030 SOUND 1,-15,121,1
20040 SOUND 1,-15,B%,1
20050 SOUND 1,-15,A%,1
20060 SOUND 1,-15,255,1
20070 SOUND 1,-15,B%,1
20080 SOUND 1,-15,A%,1
20090 SOUND 1,-15,B%,1
20100 ENDPROC
20101 :
20110 DEF PROCr
20120 FOR A%=100 TO 200 STEP 25
20130 SOUND 1,-15,A%-4,1:SOUND 1,-15,A%
,1:SOUND 1,-15,A%+4,1:SOUND 1,-15,A%+8,1
20140 SOUND 1,0,1,1
20150 NEXT
20160 ENDPROC
20170 ON ERROR OFF
20180 MODE6:IF ERR<>17 REPORT:PRINT" at
 line ";ERL
```

Continued from Page 19

```
 1290 VDU 23,247,60,32,32,0,0,0,0,0
 1300 VDU 23,248,0,56,56,16,16,16,16,0
 1310 VDU 23,249,0,56,56,20,34,68,72,0
 1320 VDU 23,250,60,0,0,0,0,0,8,100
 1330 VDU 23,251,0,28,4,7,7,3,0,0
 1340 VDU 23,252,0,0,1,1,1,7,0,0
 1350 VDU 23,253,32,32,0,0,0,0,0,0
 1360 VDU 23,254,16,32,36,44,30,0,0,0
 1370 VDU 23,255,0,0,0,194,224,224,0,0
 1380 MAN1$=CHR$(18)+CHR$(3)+CHR$(1)+CH
R$(247)+CHR$(8)+CHR$(10)+CHR$(248)+CHR$
(8)+CHR$(18)+CHR$(3)+CHR$(7)+CHR$(247)+
CHR$(8)+CHR$(11)+CHR$(244)+CHR$(8)+CHR$
(18)+CHR$(3)+CHR$(5)+CHR$(246)
 1390 MAN2$=CHR$(18)+CHR$(3)+CHR$(1)+CH
R$(247)+CHR$(8)+CHR$(10)+CHR$(249)+CHR$
(8)+CHR$(18)+CHR$(3)+CHR$(7)+CHR$(250)+
CHR$(8)+CHR$(11)+CHR$(244)+CHR$(8)+CHR$
(18)+CHR$(3)+CHR$(5)+CHR$(246)
 1400 MAN3$=CHR$(18)+CHR$(3)+CHR$(1)+CH
R$(251)+CHR$(252)+CHR$(8)+CHR$(8)+CHR$(
18)+CHR$(3)+CHR$(7)+CHR$(253)+CHR$(254)
+CHR$(8)+CHR$(18)+CHR$(3)+CHR$(5)+CHR$(
255)
 1410 MAN$=MAN1$:X%=0:I%=7:y%=0
 1420 ENDPROC
 1430 :
 1440 DEF PROCscreen
 1450 VDU 19,0,4;0;19,15,4;0;19,14,4;0;
19,13,4;0;19,3,4;0;
 1460 PRINT TAB(7,1)"PLEASE"TAB(8,2)"WA
IT"
 1470 FOR Q=0 TO 39
 1480 READ C,Z,X,Y:GCOL 0,C:PLOT Z,X,Y:
NEXT
 1490 GCOL 0,1:A=1/(1000*2^.5)
 1500 FOR X=-1000 TO 1000 STEP 50
 1510 Y=((0.1/A)*(EXP(A*X)+EXP(-A*X)))-
(200000*A)
 1520 IF X=-1000:MOVE 140,Y/2+580
 1530 DRAW X/2+640,Y/2+618
 1540 IF X=1000:DRAW 1140,Y/2+580
 1550 NEXT
 1560 VDU 5,19,15,1;0;19,14,6;0;19,2,1;
0;19,4,7;0;19,6,2;0;19,13,2;0;19,3,3;0;
19,5,2;0;19,9,7;0;19,10,4;0;
 1570 GCOL 0,9
 1580 MOVE128,64:VDU242,242,32,242,242,
32,242,242,32,242,242,32,242,32,242,243
 1590 MOVE64,32:VDU32,243,32,242,243,32
,243,32,243,243,32,32,243,243,32,243,32
 1600 GCOL 0,10
 1610 MOVE 64,32:VDU 32,243,242,32,242,
32,243,243,32,242,32,242,242,32
 1620 MOVE 129,64:VDU 242,32,242,243,32
,242,243,242,32,242,243,32,242,32,242
 1630 MOVE X%,716:PRINT MAN$
 1640 VDU 28,3,29,16,12,24,192;96;1024;
640;
 1650 ENDPROC
 1660 :
 1670 DATA 15,4,0,0,15,4,190,64,15,85,0
,608,15,85,190,640,15,4,1280,0,15,4,108
8,64,15,85,1280,608,15,85,1088,640,3,4,
0,608,3,4,192,640,3,85,0,640,3,85,384,7
36,3,85,0,736,3,4,1280,608,3,4,1088,640
 1680 DATA 3,85,1280,640,3,85,896,736,3
,85,1280,736,3,4,864,896,3,4,440,896,3,
85,768,832,3,85,480,864,13,4,384,736,13
,4,0,736,13,85,512,832,13,85,0,992,13,8
5,320,992,13,4,896,736,13,4,1280,736,13
,85,768,832,13,85,1280,992,13,85,1024,992
 1690 DATA 14,4,0,992,14,4,0,1024,14,85
,1280,992,14,85,1280,1024,14,4,320,992,
14,4,1024,992,14,85,440,896,14,85,864,896
```

Subscriptions and Software Help Line
St Albans (0727) 60263
Manned Mon-Fri 1pm-4pm.

BACK ISSUES AND SUBSCRIPTIONS

BACK ISSUES (Members only)

All back issues will be kept in print (from November 1983). Send 90p per issue PLUS an A5 SAE to the subscriptions address. Back copies of BEEBUG are available to ELBUG members at this same price. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the advertising supplements are not supplied with back issues.

Subscription and Software Address

ELBUG
PO BOX 109
High Wycombe
Bucks

SUBSCRIPTIONS

Send all applications for membership, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:

U.K.
£5.90 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
Eire and Europe
Membership £16 for one year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

SOFTWARE (Members only)

This is available from the software address.

MAGAZINE CONTRIBUTIONS AND TECHNICAL QUERIES

Please send all contributions and technical queries to the editorial address opposite. All contributions published in the magazine will be paid for at the rate of £25 per page.

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

ELBUG
PO Box 50
St Albans
Herts

ELBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Production Editor: Phyllida Vanstone.
Technical Assistants David Fell, Nigel Harris and Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to Sheridan Williams, and Adrian Calcraft for assistance with this issue.

ELBUG MAGAZINE CASSETTES

CONTENTS LIST

Volume 1 Number 1 (November 1983)
Munch-Man - a Snapper type of game, Sound Wizard for designing and experimenting with sound envelopes, Graphics example program, a utility for producing double height height characters, Highlo Card Game, a colourful Union Jack display, A Keyset program for setting up the function keys, and the exciting Hedgehog game.

Volume 1 Number 2 (December 1983)
Return of the Diamond - a fascinating adventure game, a utility for displaying 3D lettering, an interesting visual display called Square Dance, ASTAAD - a versatile computer aided design program, a musical Christmas card, Robot Attack game, a Graphics example program, Santa's Parcels game, a utility to rescue 'Bad Programs', and a fast moving football game.

Volume 1 Number 3 (January/February 1984)
Mars Lander game, a program for rotating, enlarging and reducing 3D objects, examples of Electron Graphics (3), a utility for designing a new character set, Reversi board game, a changing visual pattern based on ellipses, a utility for listing 'Bad Programs', and a challenging Dive Bomber game.

Volume 1 Number 4 (March 1984)
Killer Dice game, The Spider and the Fly - an amusing visual display, further examples of Electron Graphics (7), Moving Chequer Board display, a versatile editor for developing sound envelopes, and the superb Block Blitz game.

Volume 1 Number 5 (April 1984)
Invasion of the Aliens game, more examples of Electron Graphics (6), a short utility for saving screen displays, a simulation of continually changing fabric patterns, a classic Dominoes game, a versatile Utility Editor for Basic programmers, and the exhilarating Elevasion game.

Volume 1 Number 6 (May 1984)
Hunt the Numbers game, Invisible Alarm Clock, a Selective Renumber utility for Basic programs, ASTAAD2 - the original CAD program extended, Graphics example programs (3), Lunar Escape game, Dancing Lines - an interesting visual display, and Four in a Row game.

Volume 1 Number 7 (June 1984)
Flip Flap Game, Screen Freezer utility, routines for Expanding and Rotating Characters, Cursor Keys demonstration, an interesting Stock Market Game, an action packed Galactic Invasion Game, and Niagara, the winning Electron entry in the March Brainteaser competition.

See back cover of ELBUG for full details on prices, subscription rates and ordering.

ELBUG MAGAZINE CASSETTE

To save wear and tear on fingers and brain, we offer, each month, a cassette of the programs featured in the latest edition of ELBUG. The first program on each tape is a menu program, detailing the tape's contents, and allowing the selection of individual programs. The tapes are produced to a high technical standard by the process used for the BEEBUGSOFT range of titles.

Magazine cassettes have been produced for each issue of ELBUG from Volume 1 Number 1 onwards and are all available from stock, priced £3.00 each inclusive of VAT. See below for ordering information.

This months cassette (Vol.1 No.7) includes:

Flip Flap Game, Screen Freezer utility, routines for Expanding and Rotating Characters, Cursor Keys demonstration, an interesting Stock Market Game, an action packed Galactic Invasion Game, and Niagara, the winning Electron entry in the March Brainteaser competition.

MAGAZINE CASSETTE SUBSCRIPTION

We are also able to offer ELBUG members subscription to the magazine cassette, this gives the added advantage of receiving the cassette at around the same time as the magazine each month. Subscriptions may either be for a period of 1 year or 6 months, however for an introductory period we are also offering a trial 3 months subscription. (NOTE Magazine cassettes are produced 10 times each year).

If required, subscriptions may be backdated as far as Volume 1 Number 1, so when applying please write to the address below quoting your membership number and the issue from which you would like your subscription to start.

MAGAZINE CASSETTE ORDERING INFORMATION

Individual ELBUG Magazine Cassettes £ 3.00
P & P: Please add 50p for the first and 30p for each subsequent cassette.
Overseas orders: Calculate the UK price including post, then deduct 15% VAT and add £1 per item.

Magazine Cassette Subscription

1 YEAR (10 issues)	£33.00 Incl.....	O'SEAS	£39.00	No VAT payable
6 MONTHS(5 issues)	£17.00 Incl.....	O'SEAS	£20.00	No VAT payable
3 MONTHS(3 issues)	£10.00 Incl.....	O'SEAS	£13.00	No VAT payable

Please be sure to specify that you require subscription to the ELBUG magazine cassette, and enclose your membersip number with a cheque made payable to BEEBUGSOFT.
Send to..
ELBUG Magazine Cassette, BEEBUGSOFT, PO Box 109, High Wycombe, HP10 8HQ.

Printed in England by Staples Printers St Albans Limited at The Priory Press.